Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Sindre Stenen Blakseth

Introducing CoSTA: A Deep Neural Network Enabled Approach to Improving Physics-Based Numerical Simulations

Master's thesis in Applied Physics and Mathematics
Supervisor: Trond Kvamsdal, Adil Rasheed and Tor Nordam
June 2021

NTNU
Norwegian University of
Science and Technology

Sindre Stenen Blakseth

Introducing CoSTA: A Deep Neural Network Enabled Approach to Improving Physics-Based Numerical Simulations

Master’s thesis in Applied Physics and Mathematics
Supervisor: Trond Kvamsdal, Adil Rasheed and Tor Nordam
June 2021

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Physics

Abstract

Hybrid analysis and modelling (HAM) is an emerging modelling paradigm where physics-based modelling (PBM) and data-driven modelling (DDM) are combined with the aim of creating models that are generalizable, trustworthy, accurate, computationally efficient and self-evolving. In this thesis, we introduce, justify and demonstrate the Corrective Source Term Approach (CoSTA), which is a novel generalization of earlier work within the HAM paradigm. The crux of CoSTA is to augment the governing equation of a physics-based model with a corrective source term. The corrective source term is designed to correct any error in the original, non-augmented PBM, and can be learnt using data-driven techniques such as deep neural networks. We conduct a series of numerical experiments on one- and two-dimensional heat transfer problems, and find that CoSTA significantly outperforms comparable PBM and DDM models in terms of accuracy – often reducing predictive errors by several orders of magnitude. We also find that CoSTA facilitates the development of models with excellent generalizability. Additionally, we demonstrate how the learnt corrective source term can be analysed within a physics-based framework, thereby adding a level of explainability not found in pure DDM. In addition to increasing explainability, such analyses can also be used for automatic performance monitoring. Thus, we believe that CoSTA can push data-driven techniques to enter high-stakes applications previously reserved for pure PBM.

While our experimental results are mainly centered on heat transfer problems, CoSTA is a completely general approach which can be used for modelling any deterministic system. Furthermore, CoSTA does not impose any restrictions on what kind of data-driven techniques can be used to learn the corrective source term. Due to its flexible but solid theoretical foundation, CoSTA can therefore be used in a wide variety of applications, and will be able to leverage future advances in both PBM and DDM. Among possible areas of application within Norwegian industry and research communities, we highlight aluminium production, off-shore wind and flow assurance (in the context of e.g. hydrocarbon or CO_2 transport) as particularly relevant examples.

Sammendrag

Hybrid analyse og modellering (HAM) er et fremvoksende modelleringsparadigme hvor fysikkbasert modellering (FBM) og datadreven modellering (DDM) kombineres for å utvikle modeller som er generaliserbare, pålitelige, nøyaktige, ressurseffektive og selvutviklende. I denne masteroppgaven introduserer og demonstrerer vi CoSTA (fra engelsk *Corrective Source Term Approach*), som er en innovativ generalisering av tidligere arbeid innenfor HAM-paradigmet. Vi presenterer også det teoretiske grunnlaget for CoSTA, hvor hovedprinsippet er å utvide en eksisterende FBM med et korrigerende kildeledd. Dette kildeleddet, som er definert slik at det korrigerer enhver feil i den opprinnelige FBMen, kan læres ved hjelp av datadrevne modelleringsteknikker som dyp læring med nevrale nettverk.

Vi har gjennomført en rekke numeriske eksperimenter på en- og todimensjonal varmeledning hvor CoSTA predikerer temperatureprofiler som er opptil flere størrelsesordener mer nøyaktige enn prediksjonene til sammenliknbare FBMer og DDMer. Av våre eksperimenter ser vi også at CoSTA fasiliterer utvikling av modeller som generaliserer godt. Videre demonstrerer vi hvordan det korrigerende kildeleddet kan analyseres innenfor et fysikkbasert rammeverk. Dette medfører fortolkningsmuligheter som ikke har noen parallell innen ren DDM. I tillegg kan analysemetodene vi presenterer også brukes til automatisk ytelseskontroll. Disse faktorene styrker påliteligheten til CoSTA sammenliknet med ren DDM. Tatt i betraktning at CoSTA benytter DDM-teknikker, mener vi at CoSTA dermed kan bidra til å øke relevansen av DDM innenfor bruksområder hvor FBM tradisjonelt har vært foretrukket. Dette er særlig aktuelt innen bruksområder med høy finansiell eller sikkerhetsmessig risiko, hvor pålitelighet er av særskilt betydning.

Selv om våre eksperimenter fokuserer på varmeledning, kan CoSTA også brukes til å modellere enhver annen deterministisk prosess. Dessuten legger ikke CoSTA noen begrensninger på hvilke DDM-teknikker som kan brukes for å lære det korrigerende kildeleddet. Denne fleksibiliteten gjør at CoSTA kan dra nytte av fremtidige innovasjoner innen både FBM og DDM. Videre medfører den også at CoSTA er relevant innenfor en lang rekke varierte bruksområder. Aluminiumsproduksjon, havvind og strømningsflyt i rørledninger (som er relevant i forbindelse med f.eks. olje- og gassproduksjon og karbonfangt og -lagring) kan trekkes frem som potensielle bruksområder innen norsk industri og forskning.

Preface

This thesis concludes my Master of Science degree in Applied Physics and Mathematics at the Norwegian University of Science and Technology (NTNU). Working on this thesis has been both interesting and rewarding, and I feel that the end result is a worthy ending to my five years as a student at NTNU. I am especially contempt with my thesis topic, as I believe that data-driven techniques have a great potential for use in natural sciences and engineering which is yet to be unlocked. It is my hope and aspiration that this thesis can prove useful to students and researchers looking to combing data-driven techniques with physics-based modelling.

During my thesis work, I have been supervised by Professor Trond Kvamsdal (Department of Mathematical Sciences, NTNU), Professor Adil Rasheed (Department of Engineering Cybernetics, NTNU) and Associate Professor Tor Nordam (Department of Physics, NTNU), and I would like to thank all three of them for their contributions to my thesis. I am particularly grateful to Trond and Adil for suggesting the thesis topic to me, for helping me in shaping the numerical experiments and overall direction of the thesis, and for providing feedback on my writing. I would also like to thank Tor for aiding me with the formalities of the thesis.

Another person who has been instrumental in the shaping of this work is my girlfriend Alexandra Metallinou Log. Our day-to-day discussions have been an invaluable contribution for which I am very grateful. I also want to express my gratitude for her assistance with the experiments described in Appendix A, for her feedback on my writing, and for the love and support I receive from her. I am also very grateful for the love and support I receive from my parents.

While working on this thesis, I have had the pleasure of adapting some of my work into an article (Blakseth et al., 2021). This article, which focuses mainly on my work on one-dimensional problems, was written by me in collaboration with my supervisors Trond and Adil, in addition to Assistant Professor Omer San (Oklahoma State University). I would like to thank the co-authors for their help in refining my work to the level where it could be submitted to a journal. A follow-up article focusing on two-dimensional problems is also in the pipelines.

I want to thank the staff at NTNU, and particularly at the Department of Physics, for offering the Applied Physics and Mathematics study program and the Applied Physics specialization. I would recommend this study program and specialization to anyone with an interest in the natural sciences. At last, I want to thank my fellow students and Linjeforeningen Nabla for their great contributions to my time as a student.

Acknowledgement: The present work is part of the research projects EXAIGON–*Explainable AI systems for gradual industry adoption* (grant no. 304843), *Hole cleaning monitoring in drilling with distributed sensors and hybrid methods* (grant no. 308823), and RaPiD–*Reciprocal Physics and Data-driven models* (grant no. 313909).

Trondheim, June 2021
Sindre Stenen Blakseth

Contents

Nomenclature

Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
BC	Boundary Condition
CD	Contact Discontinuity
CFL	Courant–Friedrichs–Levy
CoSTA	Corrective Source Term Approach
CPU	Central Processing Unit
DDM	Data-Driven Modelling
DNN	Deep Neural Network
DT	Digital Twin
ENSO	El Niño Southern Oscillation
FC	Fully Connected
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
GNN	Graph Neural Network
GPU	Graphics Processing Unit
HAM	Hybrid Analysis and Modelling
LSTM	Long Short-Term Memory Network
LxF	Lax–Friedrichs Method
MSE	Mean Squared Error
NN	Neural Network
PBM	Physics-Based Modelling
PDE	Partial Differential Equation
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROM	Reduced Order Modelling
TPU	Tensor Processing Unit
VAE	Variational Autoencoder

Scalars

α	Generic parameter	
a	Entries of matrix $\mathbb{A}$	
A	Area	m^2
β	Neuron bias	
c	Speed of sound or general wave speed	m/s
c_p	Specific heat capacity at constant pressure	J/kgK
c_V	Specific heat capacity at constant volume	J/kgK
C	General constant	
ϵ_c	Error in modelling of c	m/s
ϵ_k	Error in modelling of k	W/Km
ϵ_P	Error in modelling of P	$\mathrm{W/m}^3$
ϵ_ρ	Error in modelling of ρ	$\mathrm{kg/m}^3$
e	Specific internal energy	J/kg
E	Energy per unit volume	$\mathrm{J/m}^3$
$\mathcal{E}$	Error	
$\mathbb{E}$	Experience	
ϕ	Activated neuron output	
γ	Heat capacity ratio	
κ	Thermal diffusivity	$\mathrm{m}^2\mathrm{/s}$
k	Thermal conductivity	W/Km
λ	Theoretical order of accuracy	
λ_ν	Empirical order of accuracy at grid refinement level ν	
Λ	LeakyReLU scaling of negative inputs	
l	General counter variable	
ν	Grid refinement level	
n	Time level	
n_{hist}	Number of historic time levels used to define τ	
N_i	Number of grid cells in y-direction	
N_j	Number of grid cells in x-direction	
N_ν	Number of grid refinements in grid refinement study	
N_O	Dimensionality of general DNN output vector $\boldsymbol{O}$	
N_t	Number of time levels	
N_ω	Dimensionality of general vector $\boldsymbol{\omega}$	

Ω	General domain	
$\partial\Omega$	General domain boundary	
p	Pressure	Pa
P	Heat generation rate per unit volume	$\mathrm{W/m^3}$
$\mathbb{P}$	Performance measure	
Q	Heat	J
ρ	Density	$\mathrm{kg/m^3}$
r	Residual	
σ	Heating/cooling rate	K/s
$\hat{\sigma}$	Corrective source term (continuous formulation)	
t	Time	s
t_0	Initial time	s
t_{end}	End-time of some process.	s
Δt	Process duration (in Section 2.2.2)	s
Δt	Time step (outside Section 2.2.2)	s
T	Temperature	°C
$\mathbb{T}$	Class of tasks	
u	General variable	
U	Internal energy	J
ΔU	Change in internal energy	J
v	Velocity in x-direction	m/s
V	Volume	$\mathrm{m^3}$
∂V	Surface of volume V	$\mathrm{m^2}$
W	Work	J
x	Position along 1st coordinate axis	m
y	Position along 2nd coordinate axis	m
Δx	Length of grid cell along x-axis	m
Δy	Length of grid cell along y-axis	m
ζ	Grid refinement factor	
z	Neuron output	

Vectors and Matrices

$\mathbb{A}$	Coefficient matrix of Implicit Euler FVM on matrix form	
$\boldsymbol{b}$	Right-hand side of Implicit Euler FVM on matrix form	
$\hat{\boldsymbol{\epsilon}}_k$	HAM-predicted error in PBM modelling of k	W/Km
$\hat{\boldsymbol{\epsilon}}_P$	HAM-predicted error in PBM modelling of P	W/m^3
$\hat{\boldsymbol{\epsilon}}_\rho$	HAM-predicted error in PBM modelling of ρ	kg/m^3
$\boldsymbol{F}$	Flux vector	
$\boldsymbol{\mathcal{F}}$	Numerical flux vector	
$\boldsymbol{\eta}$	General neuron input vector	
$\boldsymbol{I}$	General DNN input vector	
$\hat{\boldsymbol{n}}$	Unit surface normal vector	
$\boldsymbol{O}$	General DNN output vector	
$\boldsymbol{q}$	Heat flux	W/m^2
$\boldsymbol{R}$	Right-hand side of spatially discretized heat equation	K/s
$\boldsymbol{\sigma}$	Discretized heating/cooling rate	K/s
$\hat{\boldsymbol{\sigma}}$	Corrective source term (discrete formulation)	
$\hat{\boldsymbol{\sigma}}_{\text{NN}}$	DNN-generated corrective source term	
$\boldsymbol{T}$	Discretized temperature	°C
$\boldsymbol{U}$	Vector of conserved variables	
$\boldsymbol{w}$	Neuron weights	
$\boldsymbol{\omega}$	General vector	
$\boldsymbol{\xi}$	Vector containing selected elements of $\hat{\boldsymbol{\sigma}}$	
$\boldsymbol{z}$	Output of DNN layer	

Superscripts

i	Iteration level
n	Time level

Subscripts

a	Left domain boundary
b	Right domain boundary
c	Bottom domain boundary
d	Top domain boundary
d	DDM prediction
discr	Related to discretization error
e	Right (eastern) boundary of control volume
h	HAM prediction
i	Grid node index in y-direction
$i + 1/2$	Cell face index in y-direction
j	Grid node index in x-direction
$j + 1/2$	Cell face index in x-direction
L	Left state
mod	Related to modelling error
ν	Grid refinement level
n	Upper (northern) boundary of control volume
NN	Predicted by a (possibly deep) NN
p	PBM prediction
ref	Exact (reference) data
R	Right state
s	Lower (southern) boundary of control volume
w	Left (western) boundary of control volume

Modifiers

$\bar{}$ (overline)	Cell-averaged quantity
$\underline{}$ (underline)	Quantity containing boundary information
$\tilde{}$	Quantity affected by some error
$\hat{}$	Corrected or corrective quantity

Functions and Operators

$\lvert \cdot \rvert$	Absolute value
$\lVert\cdot\rVert_2$	ℓ_2-norm
$\boldsymbol{\nabla}$	Gradient operator (nabla)
∇^2	Laplacian
%	Modulo operator
DNN_σ	DNN predicting a corrective source term
DNN_T	DNN predicting a temperature profile
DNN_ξ	DNN predicting a vector $\boldsymbol{\xi}$
exp	Exponential function
Φ	Activation function
f	General function
g	General function related to BCs
$\mathcal{I}$	General integrable interpolant
$\log_\zeta$	Base-ζ logarithm
μ	(Empirical) mean
max	Largest element in a set
$\mathcal{N}_\Omega$	General operator defined on domain Ω
$\mathcal{N}_{\partial\Omega}$	General operator defined on domain boundary $\partial\Omega$
$\mathcal{N}_{\mathrm{num}}$	General numerical operator
std	(Empirical) standard deviation
τ	Transition function

Sets

$\mathcal{A}_{\mathrm{train}}$	Set of α-values used for DNN training
$\mathcal{A}_{\mathrm{val}}$	Set of α-values used for DNN validation
$\mathcal{A}_{\mathrm{test}}$	Set of α-values used for model evaluation (testing)
$\mathbb{N}$	The set of (positive) natural numbers
$\mathbb{R}$	The set of real numbers

1. Introduction

1.1. Background and Motivation

In understanding and interacting with the world around us, predictive modelling techniques are paramount. Such techniques are used in a vast and diverse array of applications, including weather forecasting, construction engineering and economics, to name just a few. In all these applications, a model – which is essentially a structured collection of knowledge – is used to make predictions (e.g. of the weather tomorrow, the maximum load of a bridge or the housing prices next month). Another field where predictive modelling plays an important role is that of modern digital twin (DT) technologies (Boschert and Rosen, 2016; Rasheed et al., 2020). Digital twins, which are defined by Rasheed et al. (2020) as virtual representations of physical assets enabled through data and simulators, have seen a surge in popularity following the recent wave of digitalization. Among their numerous applications, we find industrial manufacturing (Rosen et al., 2015; Stanko and Stommel, 2020; Schleich et al., 2017), systems engineering (Madni et al., 2019), product lifetime management (Tao et al., 2018) and aerospace engineering (Shafto et al., 2012; Negri et al., 2017). In their work concerning the modelling aspect of DTs, San et al. (2021) have identified the following key modelling characteristics:

1. Generalizability – the ability of the model to solve a variety of problems without problem-specific fine-tuning.

2. Trustworthiness – the extent to which the model can be explained and analyzed by human users and/or automated systems, e.g. for the purpose of monitoring the performance of the model.

3. Computational efficiency and accuracy – the ability of the model to match ground truth data while keeping computational cost to a minimum.

4. Self-adaptation – the ability of the model to learn and evolve as new situations are encountered, even after the model has been deployed.

While San et al. (2021) consider these characteristics within the context of DTs, they are certainly desirable in other contexts as well. Notable examples include autonomous systems, flow assurance and off-shore wind modelling. Thus, when developing any new predictive modelling technique, it is highly desirable for the technique to possess all of the four characteristics listed above.

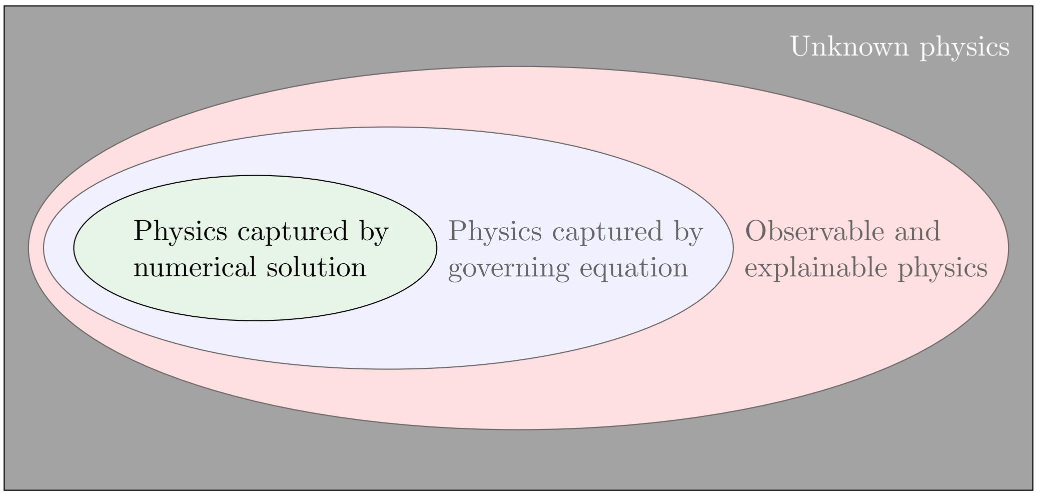
Historically, most predictive modelling techniques could be classified as either physics-based modelling (PBM) or data-driven modelling (DDM). In addition, the new modelling paradigm *hybrid analysis and modelling* (HAM), in which PBM and DDM are combined in a single hybrid model, has recently been gaining traction (San et al., 2021). These three kinds of modelling, and some examples of their applications, are described briefly below.

Physics-Based Modelling: For any real-world system, physics-based modelling aims to explain the system's behaviour using existing knowledge of observable and explainable physics (illustrated by the red ellipse in Figure 1.1a). Thus, PBM is ignorant of any unknown physics, i.e. physics that is presently unrecognized or unexplained (the dark background in Figure 1.1a). Using the first principles of known physics, PBM requires the derivation of a governing equation. For example, when considering heat transfer, known fundamentals such as the first law of thermodynamics can be used to derive a governing equation describing the temporal and spatial development of a system's temperature. Typically, these derivations require one or more assumptions such that only partial physics is captured by the governing equation (the blue ellipse in Figure 1.1a). Furthermore, governing equations are often difficult, if not impossible, to solve analytically. For the physics-based model to yield any quantitative prediction, the governing equation must then be solved with a numerical solver, the use of which can be very computationally expensive. In addition, approximations made within a solver, such as e.g. approximating a derivative with a finite difference approximation, can result in further loss of physics. Thus, the physics captured by a PBM (the green ellipse in Figure 1.1a) is generally only part of the full physics governing the studied system. Furthermore, PBMs are generally static, meaning that they do not automatically adapt to new situations after model deployment.

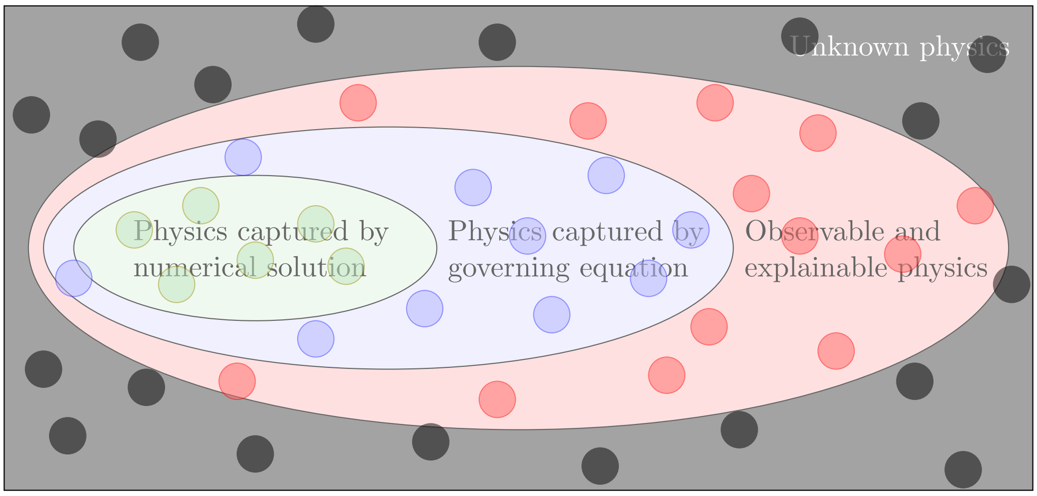
Despite the weaknesses outlined above, PBM has seen wide-spread use in a vast array of applications. Examples include weather forecasting (Müller et al., 2017), oil spill modelling (Nordam et al., 2019), CO_2 transport (Munkejord et al., 2016; Log et al., 2021) and architectural engineering (Tuan and Shang, 2014), to name just a few. Additionally, an extensive review of PBM within DT applications can be found in (Rasheed et al., 2020). The popularity of PBM largely stems from its great generalizability and trustworthiness, both of which are results of its sound first-principles foundation. Furthermore, the theory of numerical mathematics can be used to provide error bounds and stability criteria for the numerical methods used in PBM. Thus, the behaviour of PBMs are generally well-understood because we know exactly which physics is included in a given model, and how well the included physics can be resolved by numerical methods when analytic solutions are unattainable. These characteristics are particularly valued in high-stakes industrial applications.

Data-Driven Modelling: In contrast to PBM, DDM is not limited to modelling only known and understood physics. To the contrary, data-driven modelling thrives on the notion that observational data is a manifestation of both known and unknown physics, as illustrated in Figure 1.1b. Thus, if sufficient observational data is available, DDM can learn the full physics governing a system on its own. As a growing number of industrial and scientific applications migrate from sparse-data to big-data domains, the applicability of DDM is increasing. Recent activity in DDM is also facilitated by the availability of open-source cutting-edge machine learning libraries such as PyTorch and TensorFlow, and by improvements in the quality and cost-effectiveness of computational infrastructure such as GPUs and TPUs.

In recent years, data-driven techniques – and in particular those involving deep neural networks (DNNs) – have excelled at a multitude of tasks that were long considered too challenging for computers. Notable examples include image classification (Szegedy et al., 2017), speech recognition (see Bai and Zhang (2021) for a recent review), medical diagnostics (Liu et al., 2019), image synthesis (Karras et al., 2019) and even playing the board-game Go (Silver et al., 2016). These advances have recently helped push DDM for scientific and engineering applications as well. Recent use of DDM in domains traditionally dominated by PBM includes tropical cyclone intensity estimation (Lee et al., 2021), modelling of percussive drilling (Afebu et al., 2021), and forecasting of wind (Chen et al., 2018), precipitation (Shi et al., 2015) and solar activity (Pala and Atici, 2019).

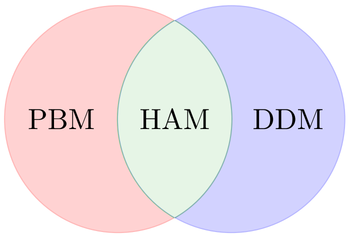

Unknown physics
Physics captured by numerical solution
Physics captured by governing equation
Observable and explainable physics

(a) Physics captured by PBM.


Unknown physics
Physics captured by numerical solution
Physics captured by governing equation
Observable and explainable physics

(b) Physics captured by DDM.

Figure 1.1.: In general, the full physics governing a system may encompass both known and unknown physics, as illustrated by the red ellipse and the black background. PBM accounts for a portion of the known physics (the green ellipse), while DDM accounts for observed physics (the small circles) no matter if it is known or unknown to physicists.



(a) HAM combines PBM and DDM in a single hybrid model.

	PBM	HAM	DDM
Generalizability	☺	☺	☹
Trustworthiness	☺	☺	☹
Computational efficiency	☹	☺	☺
Self-adaption	☹	☺	☺

(b) HAM retains the strengths of PBM and DDM while eliminating their weaknesses.

Figure 1.2.: Cartoon comparison of physics-based modelling (PBM), data-driven modelling (DDM) and hybrid analysis and modelling (HAM).

As mentioned above, DDMs are not limited by current human knowledge since they learn from observations which manifest both known and unknown physics. For this reason, DDM can offer superior accuracy in comparison to PBM, especially in scenarios where important characteristics of the system or process being modelled are unknown. Furthermore, DDMs are typically less computationally expensive than PBMs. For example, the numerical solvers often found in PBMs require large linear (or even non-linear) systems of equations to be solved for a prediction to be made. In contrast, DNN-based DDMs only require basic matrix multiplications and evaluations of simple functions. Another advantage of DDMs is that they can continue to learn from and adapt to new scenarios even after deployment – a significant improvement over the static nature of PBMs. On the downside, DDMs can be difficult to interpret due to the black-box-like nature of DNNs, and their generalizability is inherently limited due to the bias-variance trade-off. The lack of well-known and robust model stability analysis methods is another factor currently keeping DDMs away from high-stakes applications. For example, DNN-based models are vulnerable to so-called adversarial examples, as studied by e.g. Szegedy et al. (2013) and Akhtar and Mian (2018), and DNN-misbehaviour due to adversarial examples may present an unacceptable security risk in safety-critical applications.

Hybrid Analysis and Modelling: From the discussion above, it is clear that neither PBM nor DDM possess the four desirable modelling characteristics identified by San et al. (2021). However, we observe that all four characteristics are possessed by either PBM or DDM, which motivates the combined use of both PBM and DDM in a single model. The modelling paradigm *hybrid analysis and modelling* (HAM) encompasses such hybrid models, as illustrated in Figure 1.2. Rasheed et al. (2020) define HAM as a modelling approach that combines the interpretability, robust foundation and understanding of PBM with the accuracy, efficiency, and automatic pattern-identification capabilities of advanced DDM. This broad definition encompasses a wide variety of different models and hybridization approaches. In their recent survey, Willard et al. (2020) highlight a large number of techniques for combining PBM and DDM, particularly focusing on techniques involving neural networks (NNs). These include, among others, physics-guided NN cost functions, physics-guided NN architectures and residual modelling. The mentioned techniques cover a diverse range of applications such as reduced order modelling[1] (ROM), inverse modelling and solving PDEs (see e.g. (Fan and Ying, 2020; Sun et al., 2020; Thompson and Kramer, 1994; Wan et al., 2018; Yang et al., 2020; Ruthotto and Haber, 2019) as cited by Willard et al. (2020)). Below, we provide our own brief survey of different approaches to HAM.

One important approach to HAM is to *speed up* computationally expensive PBMs using data-driven techniques. Some researchers (Lagaris et al., 1998; Sirignano and Spiliopoulos, 2018; Ranade et al., 2021) explore the possibility of using NNs to replace numerical solvers altogether. Others focus on improving pre-existing solvers, for example by using NNs to learn improved coefficients in general time-stepping schemes such as implicit multi-step methods (Mishra, 2018) or high-order Runge-Kutta schemes (Raissi et al., 2019). Some particularly interesting research in this direction has been conducted by Hsieh et al. (2019), who define an NN-generated correction term for iterative solvers which shows remarkable generalization while retaining favourable properties of the unmodified solvers. They report that their NN-corrected solvers use less than half the computation time of unmodified state-of-the-art solvers. A different approach to speeding up PBMs with HAM is to use data-driven techniques in the development of ROMs (Pawar et al., 2020a,b; Ahmed et al., 2019). HAM also facilitates the combination of

[1]Quarteroni et al. (2015) provide a comprehensive introduction to reduced order modelling.

ROMs with full-order models for applications where a ROM does not sufficiently resolve all physics on its own (Pawar et al., 2021a).

Another fruitful approach to HAM is to improve the *accuracy* of well-known, robust PBMs using data-driven techniques. For example, Wang et al. (2017) and Wu et al. (2018) use random forest regression to compute an improved Reynolds stress tensor for use in simulations based on the Reynolds-averaged Navier–Stokes equations. Also in the field of computational fluid dynamics, Maulik et al. (2019) and Sirignano et al. (2020) demonstrate the use of DNN-generated closures for large eddy simulations, and report improved accuracy in comparison to benchmark physics-based closures such as the Smagorinsky model (Smagorinsky, 1963). Other works where DNNs are used to directly capture modelling errors of the PBM include those by Hanna et al. (2020) and Pathak et al. (2020). In their work, Pathak et al. (2020) additionally consider another HAM approach, which is to increase the *resolution* of numerical PBM solutions using data-driven techniques. This line of work is also followed by Tran et al. (2020) as they increase the resolution of wind fields using an adversarial framework. An adversarial framework is also employed by Xie et al. (2018) who use a temporal discriminator to ensure temporal coherence in super-resolved simulations of turbulent flow.

In addition to the different HAM approaches listed above, HAM can also be used for parameter discovery. In parameter discovery applications, it is assumed that the governing equation of the system at hand is known, but that some parameters of the system are unknown or have some associated uncertainty. Pawar et al. (2021b) show how a simple DNN can be used to improve a simplified PBM of aerodynamic coefficients, while Raissi and colleagues successfully demonstrate how parameters of a governing equation can be learnt directly from data using Gaussian priors (Raissi and Karniadakis, 2018) or NNs (Raissi et al., 2019). Using a conceptually similar approach, Pun et al. (2019) use an ensemble of NNs to predict an interatomic potential for atomistic modelling of aluminium. Also in this line of work, Vaddireddy et al. (2020) successfully apply symbolic regression with gene expression programming and sequential threshold ridge regression for parameter discovery.

From the work highlighted above, it is clear that HAM enables the leveraging of data-driven techniques for improving both the computational efficiency and accuracy of physics-based models. This conclusion is supported by Willard et al. (2020), who identify data-driven techniques as being particularly useful in solving two classes of problems: 1) problems where current computational resources are insufficient for obtaining results of the desired accuracy, and 2) problems where no complete set of governing equations is known. That is to say, combining PBM with DDM into a hybrid model is helpful when stand-alone PBM lacks computational efficiency or accuracy. The use of DDM also makes such hybrid models inherently self-adaptive. Furthermore, an important guideline in HAM is to utilize relevant physics knowledge to the greatest extent possible, such that hybrid models retain the generalizability and trustworthiness of their underlying PBM. Thus, by combining PBM and DDM in a HAM framework, we can create models which satisfy all four modelling characteristics identified by San et al. (2021). This observation is the main motivation behind the study of HAM in present work.

1.2. Contribution and Research Objectives

The main theoretical contribution of the present work is the formal introduction and theoretical justification of the Corrective Source Term Approach (CoSTA) to HAM. The key concept of CoSTA is to perturb the governing equation of a PBM using a corrective source term generated by a data-driven model such as a deep neural network. The primary purpose of the corrective source term is to improve the accuracy of the PBM. This concept builds on common practice in the field of computational fluid dynamics, where *physics-based* corrective source terms have seen widespread use. Examples include the use of closure relations in large eddy simulations (Smagorinsky, 1963) and the use of non-conservative terms to account for 2D effects in 1D fluid flow simulations (Brown et al., 2015; Log et al., 2021). The possibility of using *data-driven* corrective source terms has been explored in recent research by Maulik et al. (2019) and Sirignano et al. (2020). However, these works consider only application to large eddy simulations, and are limited in their discussion of the generalizability and interpretability of the learnt source term. The CoSTA framework proposed here can be viewed as a generalization of the work by Maulik et al. (2019) and Sirignano et al. (2020), and is a fully general approach which can be used to improve the accuracy of *any* deterministic PBM. We also provide detailed discussions on the generalizability and trustworthiness of CoSTA-based HAM models.

Another important contribution of the present work is the demonstration of CoSTA in numerical experiments on unsteady heat transfer problems. Using these experiments, we aim to answer the following research questions:

1. How does the predictive accuracy of a hybrid model using CoSTA compare to the accuracy of stand-alone PBM and DDM?

2. How does the generalizability of a hybrid model using CoSTA compare to the generalizability of stand-alone PBM and DDM?

3. Are predictions made by a CoSTA-based HAM model trustworthy?

These questions are all centered on the same topic: does CoSTA facilitate the development of models which satisfy the four characteristics highlighted by San et al. (2021)? Note, however, that we do not aim to provide a perfect predictive model for any given application. To the contrary, we have selected basic PBMs and DDMs and simple physical problems in an effort to ensure that the technical details do not obscure the discussion on the general CoSTA framework.

To summarize, we enumerate the contributions of the present work as follows:

1. We formally define and justify the Corrective Source Term Approach (CoSTA) based on a general PBM.

2. We demonstrate the benefits of CoSTA in comparison to stand-alone PBM or DDM with regards to accuracy and generalizability. We particularly highlight our explicit consideration of generalizability to extrapolation scenarios.

3. We provide a detailed discussion on the trustworthiness of CoSTA-based HAM models, including a demonstration of how the learnt corrective source term can be interpreted in a physics context.

All code used to produce the results of the main text was written by the author in Python 3.6. The author wishes to highlight the use of the Python libraries PyTorch (Paszke et al., 2019), NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020) and Matplotlib (Hunter, 2007). The code used to perform the physics-based modelling described in Appendix A is based on an unpublished Fortran implementation by A. M. Log, and was translated into Python by the author. The exact solutions in the same appendix are courtesy of A. M. Log using the exact Riemann solver from the NUMERICA library (Toro, 1999).

1.3. Structure of the Thesis

This thesis has five chapters, the first of which is the current introduction. Here, we have motivated the present work (Section 1.1), presented our research questions and briefly described our most important contributions (Section 1.2).

In chapter 2, we present theory related to predictive modelling. The importance of modelling heat transfer is presented first, in Section 2.1, while PBM and DDM are covered in Sections 2.2 and 2.3, respectively. In the latter sections, we consider PBM and DDM both in a general context, and in the context of heat transfer modelling.

Chapter 3 contains the major theoretical contribution of the present work, which is the formal introduction and justification of CoSTA. The benefit of using a corrective source term is explained in a general context in Section 3.1, and in the context of heat transfer modelling in Section 3.2. In Section 3.3, we explain why we propose to learn the corrective source term using a deep neural network.

The numerical experiments where we compare CoSTA to stand-alone PBM and DDM are presented and discussed in Chapter 4. Our experimental setup is described in Section 4.1, while our experiments on one- and two-dimensional heat transfer problems are covered in Sections 4.2 and 4.3, respectively. The predictive uncertainty of data-driven and hybrid models is investigated in Section 4.4, and the effects of grid refinement on PBM, DDM and CoSTA models are explored in Section 4.5. In Section 4.6, we provide an extensive discussion on how the corrective source term can be interpreted in a physics context. Section 4.7 contains a summary of Chapter 4 where our empirical findings are related to the research questions stated in Section 1.2.

Finally, in Chapter 5, the present work is concluded. The theoretical contributions and empirical findings of the present work are summarized in Section 5.1, while the most important directions of further research are addressed in Section 5.2.

2. Theory

This chapter provides the theory required for performing physics-based modelling (PBM) and data-driven modelling (DDM) of one- and two-dimensional time-dependent heat transfer problems. We also discuss why heat transfer modelling is important. The discussion on the importance of heat transfer modelling is presented first, in Section 2.1. Thereafter, PBM and DDM are discussed in Sections 2.2 and 2.3, respectively. In each of these two sections, we begin by covering the history, basic concepts and important applications of the modelling approach covered in that section. In the latter parts of Sections 2.2 and 2.3, we describe specific PBM and DDM models and techniques, particularly focusing on heat transfer modelling.

2.1. The Importance of Heat Transfer Modelling

Naturally, heat transfer modelling is relevant in any application where heat conductance has significant impact on the (local or global) behaviour of the system being studied. This is a common situation in a large number of industrial engineering and design applications. Trivial examples include metal casting, where heat transfer modelling is imperative to ensure that the molds can handle the heat from the molten metal; computer design, where sufficient heat transfer is essential in avoiding component malfunction due to CPU overheating; and even cookware design, where care must be taken to avoid heat transfer into the handles.[1] Other examples include calculating thermal stresses in materials and constructions (Hunt and Cooke, 1975; Hetnarski and Eslami, 2009), optimal design through minimizing heat loss (Münch et al., 2008), development of thermal cloaking techniques[2] (Ma et al., 2013) and modelling of refrigeration systems (Risser et al., 2010).

Heat transfer modelling is also relevant to applications where heat conduction is not the most prominent physical process. An example of this is pipeline depressurization, where advection is the physical process that governs the behaviour of the system at large. However, for increased accuracy, one may still want to model heat transfer, e.g. from the pipe itself to the fluid inside. One can then use a heat transfer model like the unsteady heat equation (cf. Section 2.2.2) to approximate the heat flux into the fluid, and then add this heat flux as a source term in the advection equation (Munkejord and Hammer, 2015).

Outside of industrial applications, heat transfer modelling is important in climate modelling, where thermal conduction through the soil plays an important role in modelling surface temperature and surface energy balance (Bhumralkar, 1975; Pitman, 2003). Thermal conduction has also been identified as a significant factor in the study of glacier energy-balance modelling (Pellicciotti et al., 2009). In a more exotic application, Roger et al. (2016) consider an analogue between the unsteady heat equation and the Newton–Schrödinger equation for experimental modelling of boson stars using lasers.

[1]The author's empirical experience has shown that the heat transfer modelling in cookware design is oftentimes not performed (to a sufficient extent) in practice. However, this has made the author all-the-more convinced that heat transfer modelling is important in this application.

[2]Such techniques can e.g. be used for protecting heat-sensitive components from heat-generating components in complex electrical systems

In addition to being highly useful in its own right, heat transfer modelling is also important because its governing equation – the heat equation – is mathematically equivalent to many other important equations, as hinted at by the last example above. Of these analogies, diffusion is arguably the most important one. Indeed, heat transfer, as described by the heat equation, may be viewed as a special case of diffusion where the diffusing quantity is thermal energy. This is why such heat transfer phenomena are sometimes referred to as heat *diffusion*. Narasimhan (1999) gives an interesting discussion on the history of the analogy between heat transfer and diffusion, while Jost (2007) considers the analogy in great detail from a mathematical perspective. Jost (2007) also extends the analogy to reaction-diffusion processes, which are widely used in chemistry. Other analogies, such as hydraulic and electrodynamic phenomena, and their possible applications, are described in the work Bhattacharyya (1965) and references therein. The heat equation also has an analogy in quantum mechanics, as the Schrödinger equation (see (Hemmer, 2015a) for an introduction), which governs the wave function of a quantum mechanical system, is mathematically equivalent to the heat equation. Additionally, in financial sciences, the Black–Scholes equation used to model option pricing can be transformed into the heat equation (see (Van der Wijst, 2013, chapter 8) and references therein). Furthermore, for time-independent problems, the heat equation is a prototypical example of the Poisson equation, which is widely studied in e.g. (Quarteroni, 2014). Among the numerous applications of the Poisson equation, we find electro- and magnetostatics (Griffiths, 2017), Newtonian gravitation,[3] and the incompressible Navier–Stokes equations (Tannehill et al., 1997, chapter 9). Thus, it is clear that modelling techniques for heat transfer problems are highly useful in a vast array of applications, including applications where the connection to heat transfer problems is not immediately obvious. This broad relevance is an important motivation for our study of heat transfer problems.

Another important motivational factor for the present work is the great availability of temperature data. Temperature is known to affect easily measurable material properties such as volume and resistivity (Lillestøl et al., 2015; Kittel, 2005), and this facilitates the development of simple, inexpensive temperature measurement devices like thermometers, bimetallic devices and thermistors. Furthermore, temperature gradients are known to give rise to thermoelectric effects like the Seebeck effect, which facilitates additional sensor technologies like thermocouples. Temperature can also be measured in a completely non-intrusive way using thermal imaging techniques, since the heat radiated by an object is directly related to its temperature in terms of the radiation's energy density and frequency spectrum. With all these methods for temperature measurements, it is generally possible to generate ample amounts of temperature data for use with data-driven techniques, either as stand-alone DDM or as part of a HAM model.

In addition to its availability, temperature data also benefits from being informative about the state of the studied system. For example, in medicine, a fever is often a sign of disease. High temperatures may also be a sign of component malfunction in inanimate systems like computers or combustion engines. Overall, abrupt or otherwise unexpected changes in a system's temperature are often associated with some kind of an anomaly in the system, such as a component that is beginning to fail. Temperature measurements (especially when coupled with subsequent modelling) can thereby help in developing cost-effective anomaly detectors. These can be used e.g. in autonomous systems or digital twins, for example to make decisions regarding system maintenance. As such,

[3]This follows from the fact that Newton's law of gravity is mathematically equivalent to Coulomb's law of electrostatic force. See e.g. (Lien and Løvhøiden, 2015, page 334) and (Griffiths, 2017, page 60) for definitions of Newton's law and Coulomb's law, respectively.

the impact of heat transfer modelling techniques will increase further as autonomous systems and digital systems see increasingly widespread use.

2.2. Physics-based Modelling

This section is devoted to a presentation of physics-based modelling (PBM). A general introduction to PBM is given in Section 2.2.1. We then focus on PBM for heat transfer problems in Sections 2.2.2 and 2.2.3. In Section 2.2.2, we present the heat equation, which is the governing equation for heat transfer problems. Subsequently, we present a numerical method for solving the heat equation in Section 2.2.3.

2.2.1. An Introduction to Physics-Based Modelling

The core concept of physics-based modelling is to model any given system using pre-existing physics knowledge. For any given modelling problem, the first step in the development of a physics-based model is to study relevant physics theory. For example, if one is tasked with modelling heat transfer, a natural place to start would be the fundamental laws of thermodynamics.[4] These provide a solid theoretical foundation on which a PBM for heat transfer can be built. The next step is to derive a governing equation using the established theoretical foundation. For heat transfer problems, the governing equation is known as the *heat equation*, and we will discuss it further in Section 2.2.2. Governing equations like the heat equation are highly useful because they accurately describe the behaviour of any system for which their underlying first-principles hold true. This results in excellent generalizability. Additionally, the physical significance of each term in a governing equation is generally well-understood, which contributes to the high trustworthiness of PBM.

Once we have obtained a governing equation for our system of interest, we typically aim to solve this equation under some conditions to predict the system's behaviour. For some equations, such as the wave equation, this can be done analytically (Feldman, 2007). However, more often than not, we have to resort to numerical methods to solve the equations. This is the case for e.g. the heat equation. In such cases, numerical mathematics provides us with a score of numerical methods which can be used to solve the equations approximately (or even exactly, if the true solution of the equation is "nice enough"[5]). Popular classes of numerical methods include finite difference methods (FDMs), finite volume methods (FVMs) and finite element methods (FEMs) (see e.g. (Thomas, 1995), (Tannehill et al., 1997) and (Quarteroni, 2014) for introductions to FDMs, FVMs and FEMs, respectively). The existence of both theoretical and empirical analysis techniques like von Neumann stability analysis and grid refinement studies, ensures that the accuracy and convergence of such methods is generally well-understood (see e.g. (Tannehill et al., 1997, chapter 3) or (LeVeque, 2002, chapter 8) for introductions to analysis techniques for FVMs). Through the use of such techniques, it is possible to compute error bounds for the approximate solutions provided by these numerical methods. Furthermore, the approximate solutions can typically be made arbitrarily accurate,[6] provided that sufficient computational resources are available. Thus, even

[4]Hemmer (2015b) gives a concise introduction to thermodynamics.

[5]For example, a first order approximation of a derivative is exact if the true solution is linear.

[6]An important exception is that of chaotic systems, whose long-term behaviour can be practically impossible to determine numerically due to amplification of rounding errors stemming from the finite numerical precision of computers (Strogatz, 2015).

if a governing equation cannot be solved analytically, we can still obtain accurate and trustworthy approximations of its solution.

An alternative to calculating a (numerical or exact) solution of the governing equation is to analyze characteristic properties of the governing equation itself. Such analyses are an important topic in non-linear dynamics, where the qualitative behaviour of a system can remain elusive even in cases where explicit solutions of the governing equations are available (Strogatz, 2015). Examples of characteristic properties include steady-state solutions, bifurcation points, attractors and solution bounds. To examine the trustworthiness of a governing equation, the characteristic properties of the governing equation can be compared to observations of the real-world system being modelled. Similarly, to examine the trustworthiness of a numerical solution of a governing equation, one can examine if the numerical solution respects the properties of the governing equation, e.g. that the numerical solutions converges to the correct steady state. Thus, the trustworthiness of both the governing equations themselves and its (possibly numerical) solutions can be evaluated *a posteriori*. This quality of PBM is especially useful in safety-critical applications where it is imperative to monitor the well-behavedness of the predictive model.

In the numerical experiments of the present work (cf. Chapter 4), we consider a variety of different unsteady heat transfer problems in one and two dimensions where we aim to compare PBM, DDM and CoSTA-based HAM. To do this, we need a PBM for unsteady heat transfer, both for use as a stand-alone model and for use in the HAM model. The next two sections are dedicated to the derivation of such a PBM, following the general PBM development process outlined in this section. The PBM's governing equation – the heat equation – is derived from known first principles in Section 2.2.2, while we discuss how to solve this equation numerically in Section 2.2.3.

2.2.2. The Heat Equation

As discussed in the previous section, physics-based modelling is based on deriving and solving some governing equation which describes the known physics of the problem at hand. For heat transfer problems, this governing equation is known as *the heat equation*. A complete derivation of the heat equation in one dimension can be found in the specialization project report (Blakseth, 2021). Here, we repeat the main points of this derivation, while also generalizing the derivation to three dimensions. Moreover, at the end of this section, we will rewrite the heat equation on several alternative forms which will be used in later chapters.

The starting point for our derivation of the heat equation is the first law of thermodynamics, which reads

$$\Delta U = Q + W. \tag{2.1}$$

For a general system undergoing some process of duration Δt, the first law of thermodynamics states that the change ΔU in the system's internal energy U is equal to the heat Q added to the system and the work W performed on the system.

By definition, no work is performed by the system or its surroundings in pure heat transfer problems, so $W = 0$. Furthermore, the heat Q can be split up into two contributions: i) heat flowing into/out from the system, and ii) heat being generated (or consumed) within the control volume. Mathematically, this can be expressed as

$$Q = \int\limits_{t}^{t+\Delta t} \left(\int\limits_{\partial V} \boldsymbol{q} \cdot (-\hat{\boldsymbol{n}}) \; \mathrm{d}A + \int\limits_{V} P \; \mathrm{d}V \right) \mathrm{d}t', \tag{2.2}$$

where V is the system's volume, ∂V is its surface, $\boldsymbol{q} \cdot (-\hat{\boldsymbol{n}})$ is the heat flux *into* the control volume across a unit surface with unit normal $\hat{\boldsymbol{n}}$, and P is the rate at which heat is generated (per unit volume) within the system. The negative sign in front of $\hat{\boldsymbol{n}}$ appears because $\hat{\boldsymbol{n}}$ is conventionally defined as pointing *out from* the control volume. We assume that the integrand of the temporal integral in Equation (2.2) is approximately constant for small Δt, such that dividing Equation (2.1) (with $W = 0$) by Δt and inserting Equation (2.2) for Q yields

$$\frac{\Delta U}{\Delta t} = -\int\limits_{\partial V} \boldsymbol{q} \cdot \hat{\boldsymbol{n}} \, \mathrm{d}A + \int\limits_{V} P \; \mathrm{d}V. \tag{2.3}$$

In the limit $\Delta t \to 0$, we then have

$$\frac{\partial U}{\partial t} = -\int\limits_{\partial V} \boldsymbol{q} \cdot \hat{\boldsymbol{n}} \, \mathrm{d}A + \int\limits_{V} P \; \mathrm{d}V. \tag{2.4}$$

To make further progress, we employ a selection of thermodynamic identities (see (Blakseth, 2021) for the details) to rewrite the temporal derivative of U as

$$\frac{\partial U}{\partial t} = \int\limits_{V} \rho c_V \frac{\partial T}{\partial t} \mathrm{d}V, \tag{2.5}$$

where ρ denotes density and c_V denotes specific heat capacity at constant volume. Furthermore, we assume that no heat is transferred across the system's boundaries as a result of advection. Then, the heat flux $\boldsymbol{q} \cdot (-\hat{\boldsymbol{n}})$ must be caused entirely be heat conduction, which is described by Fourier's law (Lillestøl et al., 2015, chapter 18). It reads

$$\boldsymbol{q} = -k \boldsymbol{\nabla} T, \tag{2.6}$$

where k denotes thermal conductivity and $\boldsymbol{\nabla} T$ is the gradient of temperature T. Inserting Equations (2.5) and (2.6) into Equation (2.4), we obtain *the heat equation:*

$$\int\limits_{V} \rho c_V \frac{\partial T}{\partial t} \mathrm{d}V = \int\limits_{\partial V} (k \boldsymbol{\nabla} T) \cdot \hat{\boldsymbol{n}} \, \mathrm{d}A + \int\limits_{V} P \; \mathrm{d}V. \tag{2.7}$$

Equation (2.7) is often referred to as the heat equation *on integral form* due to the integrals appearing in the equation.

When considering problems in one or two dimensions, the surface integral in the three-dimensional Equation (2.7) can sometimes be expressed more conveniently. For one-dimensional problems, Equation (2.7) can be rewritten as

$$\int\limits_{V} \rho c_V \frac{\partial T}{\partial t} \mathrm{d}V = \left(kA \frac{\partial T}{\partial x} \right)_e - \left(kA \frac{\partial T}{\partial x} \right)_w + \int\limits_{V} P \; \mathrm{d}V \tag{2.8}$$

if the system's eastern (right) and western (left) boundaries are perpendicular to the x-axis. We use the subscripts $_e$ and $_w$ to denote quantities evaluated at the system's eastern and western boundary, respectively. For two-dimensional problems, Equation (2.7) can instead be rewritten as

$$\int\limits_{V} \rho c_V \frac{\partial T}{\partial t} \mathrm{d}V = \left(kA \frac{\partial T}{\partial x} \right)_e - \left(kA \frac{\partial T}{\partial x} \right)_w + \left(kA \frac{\partial T}{\partial y} \right)_n - \left(kA \frac{\partial T}{\partial y} \right)_s + \int\limits_{V} P \; \mathrm{d}V \tag{2.9}$$

if we additionally require that the system's northern (upper) and southern (lower) boundaries are perpendicular to the y-axis. The subscripts $_n$ and $_s$ denote quantities evaluated at, respectively, the northern boundary and the southern boundary.

As will be demonstrated in the next section, the integral forms given above are useful for deriving numerical solvers for the heat equation. However, for certain purposes, such as calculating the heat generation rate P given the temperature T, the integral forms can be cumbersome to work with due to the integrals. In such cases, the so-called *differential forms*, which can be derived from the integral forms under certain smoothness conditions (cf. (Blakseth, 2021, Section 3.2)), are more convenient. The one-dimensional heat equation on differential form reads

$$A\rho c_V \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(kA\frac{\partial T}{\partial x}\right) + AP, \tag{2.10}$$

while its two-dimensional counterpart is

$$A\rho c_V \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(kA\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(kA\frac{\partial T}{\partial y}\right) + AP. \tag{2.11}$$

We mention that the smoothness conditions are satisfied for all temperature profiles considered in this work, such that we can use the integral and differential forms interchangeably.

We have now defined the heat equation, which is the governing equation of unsteady heat transfer, on both integral and differential form. Unfortunately, the heat equation cannot be solved analytically in general, irrespective of which form is used to formulate it. For the purposes of predictive modelling, we therefore need a method for solving the heat equation numerically. Such a method is presented in the next section.

2.2.3. The Implicit Euler Finite Volume Method

This section is dedicated to presenting the Implicit Euler[7] Finite Volume Method (FVM). In the experiments described in Chapter 4, we use this method to solve the heat equation numerically. The Implicit Euler FVM is a popular choice for solving the heat equation because it is easy to implement and numerically stable for all discretizations of the spatial and temporal domains (Tannehill et al., 1997).

Suppose now that we want to solve the heat equation on some spatial domain Ω and temporal domain $[0, t_{\text{end}}]$ for some final time t_{end} using the Implicit Euler FVM. We must then begin by prescribing an initial temperature profile and a set of boundary conditions (BCs). In this work, we only consider problems where the boundary temperatures are fixed directly. Such BCs are commonly referred to as Dirichlet BCs.

In addition to prescribing an initial condition and a set of BCs, the Implicit Euler FVM also requires us to discretize the spatial and the temporal domains. For the temporal discretization, we split the temporal domain $[0, t_{\text{end}}]$ into N_t equally spaced time levels separated by a constant time step Δt, and we use a superscript ^{n} to denote that a quantity is evaluated at the nth time level. With this notation, we thus have $t^n = n \cdot \Delta t$. When discretizing the spatial domain, we assume that $\Omega = [x_a, x_b]$ for 1D problems and $\Omega = [x_a, x_b] \times [y_c, y_d]$ for 2D problems. The spatial discretization is considered in greater detail below, first in 1D and then in 2D.

For one-dimensional problems, we split the spatial domain $[x_a, x_b]$ into N_j grid cells, each of equal length Δx. At the center of each grid cell, and at the boundary locations

[7]Also know as the Simple Implicit Method or the Laasonen Method.

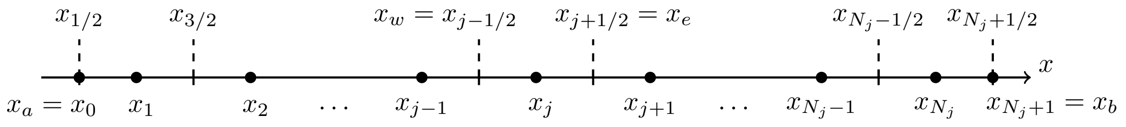


Figure 2.1.: This figure illustrates how we have discretized the spatial domain along the x-axis. Integral indices j are used to enumerate grid nodes (black dots), including the nodes at the domain boundaries. Half-integral indices $j + 1/2$ are used to enumerate cell faces (dashed lines). Note that the location of the left-most cell face coincides with the left-most grid node, such that $x_0 = x_{1/2} = x_a$. Similarly, we have $x_{N_j+1} = x_{N_j+1/2} = x_b$ at the right boundary. For 2D problems, the discretization along the y-axis is analogous. In the y-direction, we use integral indices i to enumerate grid nodes and half-integral indices $i + 1/2$ to enumerate cell faces. The boundaries are treated analogously, meaning that $y_0 = y_{1/2} = y_c$ and $y_{N_i+1} = y_{N_i+1/2} = y_d$.

x_a and x_b, we place so-called *grid nodes*, which are the locations where the numerical FVM solution will be defined. We use an integral subscript j to indicate that a quantity is evaluated at the grid node labelled j, as shown in Figure 2.1. Also illustrated in this figure are the half-integral subscripts $j + 1/2$ and $j - 1/2$ used to denote quantities evaluated at, respectively, the right and the left boundary (cell face) of the jth grid cell.

For two-dimensional problems, we treat the x-dimension just like in the 1D case, while we treat the y-dimension analogously. That is to say, we split the spatial domain $[x_a, x_b] \times [y_c, y_d]$ into $N_j \cdot N_i$ grid cells, each of equal area $\Delta x \cdot \Delta y$. Each grid node is identified by a pair of integral indices (j, i), where the first and second index denote position along the x and y dimension respectively. We use half-integral subscripts $i+1/2$ and $i - 1/2$ to denote quantities evaluated at, respectively, upper and lower cell faces, while the half-integral indices $j+1/2$ and $j-1/2$ retain their meaning from the 1D case.

Using the discretizations described above, we will now briefly explain how the Implicit Euler FVM can be derived, beginning with 1D before extending to 2D. A last point before we continue with the derivation is that, in the present work, we will only consider problems where ρ, c_V and A in Equations (2.8) and (2.9) are constant. We will also only consider FVMs where k is assumed constant.[8] For simplicity, we will therefore assume that the quantities ρ, c_V, A and k are all constant in the derivations below. Note, however, that the Implicit Euler FVM does not require such an assumption, so the derivations below can be made more general if necessary.

One Dimension

The starting point for the derivation of any FVM is the governing equation of the system at hand, written on *integral form* (LeVeque, 2002, chapter 4). For us, this means that the 1D heat equation on integral form (2.8) is our starting point. Given the discretization and assumptions outlined above, this equation can be written as

$$\int_{x_w}^{x_e} \frac{\partial T}{\partial t} \mathrm{d}x = \kappa \left(\left(\frac{\partial T}{\partial x} \right)_e - \left(\frac{\partial T}{\partial x} \right)_w \right) + \int_{x_w}^{x_e} \sigma \, \mathrm{d}x \tag{2.12}$$

for an arbitrary grid cell j whose center, right boundary and left boundary are at x_j, $x_e = x_{j+1/2}$ and $x_w = x_{j-1/2}$, respectively. Here, $\kappa = k/(\rho c_V)$ denotes thermal diffusivity and

[8]We will consider problems where the true conductivity k is varying in space and time, but, for all these problems, we synthesize modelling error by still assuming k to be constant in the FVM used.

$\sigma = P/(\rho c_V)$ can be interpreted as a heating rate (or cooling rate, if $\sigma < 0$). Note that the cross-sectional area A cancels from the equation because we assumed it to be constant.

A central concept of FVM derivations is the use of cell-averaged quantities. In the present derivation, we will make use of the cell-averaged temperature $\bar{T}$ and the cell-averaged heating rate $\bar{\sigma}$, which are defined as follows:

$$\bar{T}_j = \frac{1}{\Delta x} \int\limits_{x_{j-1/2}}^{x_{j+1/2}} T \, \mathrm{d}x, \quad \bar{\sigma}_j = \frac{1}{\Delta x} \int\limits_{x_{j-1/2}}^{x_{j+1/2}} \sigma \, \mathrm{d}x. \tag{2.13}$$

In the Implicit Euler FVM, the cell-averaged values of T and σ are assumed to be approximately equal to T and σ evaluated at the corresponding grid nodes. Additionally, the spatial derivatives are approximated using central difference approximations. Mathematically, we express these approximations as

$$\bar{T}_j \approx T_j, \quad \bar{\sigma}_j \approx \sigma_j, \quad \left(\frac{\partial T}{\partial x}\right)_{j+1/2} \approx \frac{T_{j+1} - T_j}{\Delta x_{j+1/2}} \quad \left(\frac{\partial T}{\partial x}\right)_{j-1/2} \approx \frac{T_j - T_{j-1}}{\Delta x_{j-1/2}}, \tag{2.14}$$

where $\Delta x_{j+1/2} = x_{j+1} - x_j$ and $\Delta x_{j-1/2} = x_j - x_{j-1}$.[9] With the approximations above, Equation (2.12) can be rewritten as

$$\frac{\partial T_j}{\partial t} = \frac{\kappa}{\Delta x} \left(\frac{T_{j+1} - T_j}{\Delta x_{j+1/2}} - \frac{T_j - T_{j-1}}{\Delta x_{j-1/2}} \right) + \sigma_j =: R_j. \tag{2.15}$$

We have one such equation for each of the N_j grid cells, and these N_j scalar equations can be collected into the following vector equation:

$$\frac{\partial \boldsymbol{T}(t)}{\partial t} = \boldsymbol{R}(\boldsymbol{T}(t)) \quad \text{with} \quad \boldsymbol{T}(t) = [T_1, \ldots, T_{N_j}]^T, \ \boldsymbol{R}(\boldsymbol{T}(t)) = [R_1, \ldots, R_{N_j}]^T. \tag{2.16}$$

To obtain a fully discretized system of equations, we discretize the temporal derivative such that

$$\frac{1}{\Delta t} \left(\boldsymbol{T}^{n+1} - \boldsymbol{T}^n \right) = \boldsymbol{R}\left(\boldsymbol{T}^{n+1} \right). \tag{2.17}$$

Finally, the above vector equation can be written on so-called *matrix form*

$$\mathbb{A} \boldsymbol{T}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}^n \right), \tag{2.18}$$

where $\mathbb{A}$ is a tridiagonal matrix. The non-zero elements of $\mathbb{A}$ are

$$\begin{aligned}
a_{j,j} &= 1 + \kappa \frac{\Delta t}{\Delta x} \left(\frac{1}{\Delta x_{j+1/2}} + \frac{1}{\Delta x_{j-1/2}} \right), && j = 1, \ldots, N_j, \\
a_{j,j+1} &= -\kappa \frac{\Delta t}{\Delta x} \frac{1}{\Delta x_{j+1/2}}, && j = 1, \ldots, N_j - 1, \\
a_{j-1,j} &= -\kappa \frac{\Delta t}{\Delta x} \frac{1}{\Delta x_{j-1/2}}, && j = 2, \ldots, N_j,
\end{aligned}$$

[9] $\Delta x_{j-1/2}$ is equal to $\frac{1}{2}\Delta x$ if $j = 1$, and equal to Δx otherwise, as can be seen from Figure 2.1. Similarly, $\Delta x_{j+1/2}$ is equal to $\frac{1}{2}\Delta x$ if $j = N_j$, and equal to Δx otherwise.

while the components of $\boldsymbol{b}$ are

$$\begin{aligned} b_1 &= T_1^n + \Delta t \sigma_1^{n+1} + \kappa \frac{\Delta t}{\Delta x} \frac{1}{\Delta x_{1/2}} T_a^{n+1}, \\ b_j &= T_j^n + \Delta t \sigma_j^{n+1}, & j = 2, \dots, N_j - 1, \\ b_{N_j} &= T_{N_j}^n + \Delta t \sigma_{N_j}^{n+1} + \kappa \frac{\Delta t}{\Delta x} \frac{1}{\Delta x_{N_j+1/2}} T_b^{n+1}. \end{aligned}$$

We observe that $\mathbb{A}$ is diagonally dominant, since

$$\begin{aligned} |a_{1,1}| &> \kappa \frac{1}{\Delta x_{3/2}} = |a_{1,2}| = \sum_{i \neq 1} |a_{1,i}|, \\ |a_{j,j}| &> \kappa \frac{\Delta t}{\Delta x} \left(\frac{1}{\Delta x_{j+1/2}} + \frac{1}{\Delta x_{j-1/2}} \right) = |a_{j,j+1}| + |a_{j,j-1}| = \sum_{i \neq j} |a_{j,i}|, \quad j = 2, \dots, N_j - 1, \\ |a_{N_j,N_j}| &> \kappa \frac{1}{\Delta x_{N_j-1/2}} = |a_{N_j,N_j-1}| = \sum_{i \neq N_j} |a_{N_j,i}|. \end{aligned}$$

The system (2.18) can therefore be solved using the highly efficient tridiagonal matrix algorithm. An introduction to this algorithm can be found in (Versteeg and Malalasekera, 1995, pages 157–158).

Two Dimensions

The derivation of the two-dimensional Implicit Euler FVM largely follows the same steps as the 1D derivation above. In 2D, our starting point is the 2D heat equation on integral form, which, given our assumptions of constant ρ, c_V, A and k, can be written as

$$\int_{y_s}^{y_n} \int_{x_w}^{x_e} \frac{\partial T}{\partial t} \mathrm{d}x \mathrm{d}y = \kappa \left(\left(\frac{\partial T}{\partial x} \right)_e - \left(\frac{\partial T}{\partial x} \right)_w + \left(\frac{\partial T}{\partial y} \right)_n - \left(\frac{\partial T}{\partial y} \right)_s \right) + \int_{y_s}^{y_n} \int_{x_w}^{x_e} \sigma \, \mathrm{d}x \mathrm{d}y, \quad (2.19)$$

for the grid cell centered at the grid node (x_j, y_i) and with boundaries $x_e = x_{j+1/2}$, $x_w = x_{j-1/2}$, $y_n = y_{i+1/2}$ and $y_s = y_{i-1/2}$. To discretize Equation (2.19), we use the same approximations as before[10] (cf. Equation (2.13)), in addition to the following central difference discretizations of the spatial derivatives in the y-direction:

$$\left(\frac{\partial T}{\partial y} \right)_{i+1/2} \approx \frac{T_{i+1} - T_i}{\Delta y_{i+1/2}}, \quad \left(\frac{\partial T}{\partial y} \right)_{i-1/2} \approx \frac{T_i - T_{i-1}}{\Delta y_{i-1/2}}, \quad (2.20)$$

where $\Delta y_{i+1/2} = y_{i+1} - y_i$ and $\Delta y_{i-1/2} = y_i - y_{i-1}$. With the approximations (2.13) and (2.20), Equation (2.19) can be written as

$$\frac{\partial T_{j,i}}{\partial t} = \frac{\kappa}{\Delta x \Delta y} \left(\frac{T_{j+1,i} - T_{j,i}}{\Delta x_{j+1/2}} - \frac{T_{j,i} - T_{j-1,i}}{\Delta x_{j-1/2}} + \frac{T_{j,i+1} - T_{j,i}}{\Delta y_{i+1/2}} - \frac{T_{j,i} - T_{j,i-1}}{\Delta y_{i-1/2}} \right) + \sigma_{j,i}.$$

We have $N_j \cdot N_i$ equations like the one above – one for each grid cell. These can be discretized using the same temporal discretization as we used in 1D, and the fully

[10]In 2D, the cell-averaged quantities are defined slightly differently than in 1D, since we must average over the y-dimension as well. However, we still approximate the cell-averaged quantities using the corresponding grid node values as in 1D.

discretized equations can be collected into a single vector equation. Finally, the vector equation can be written on the same matrix form as its 1D counterpart, namely

$$\mathbb{A}\boldsymbol{T}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}^{n}\right). \tag{2.21}$$

However, $\mathbb{A}$, $\boldsymbol{T}$ and $\boldsymbol{b}$ are defined differently here than in Equation (2.18). The matrix $\mathbb{A}$ is now a banded $(N_jN_i \times N_jN_i)$-matrix with five non-zero diagonals, while $\boldsymbol{T}$ and $\boldsymbol{b}$ are now N_jN_i-dimensional vectors. Their precise definitions are quite voluminous in the 2D case, and are therefore deferred to Appendix G. Since $\mathbb{A}$ is no longer tridiagonal, we cannot use the tridiagonal matrix algorithm to solve the system (2.21). However, a wide variety of other solvers are applicable. For simulations with a large number of grid points, we recommend the use of solvers that are specialized for sparse, banded coefficient matrices. However, for the purposes of our 2D numerical experiments (cf. Section 4.3), general-purpose solvers also suffice.[11]

2.3. Data-Driven Modelling

In this section, we shift our attention to data-driven modelling (DDM). A general introduction to DDM is given in Section 2.3.1, while Section 2.3.2 is devoted to a discussion on DDM for heat transfer problems. The latter section is where we explain and justify the DDM used in the numerical experiments of Chapter 4. Finally, in Section 2.3.3, we cover some basic concepts of deep neural networks, which are commonly used in DDM.

2.3.1. An Introduction to Data-Driven Modelling

When we study a system using DDM, we model the system using observational data only. This means that we do not take into account any pre-existing knowledge of the system at hand, which is a dual-edged sword. The downside is that we have to relearn whatever useful knowledge we already have. However, on the positive side, it also means that we are not limited by our current knowledge, which may be incomplete (or even incorrect). In some sense, this characteristic of DDM is reminiscent of early physics research. For example, early astronomers like Johannes Kepler did not know *why* the planets of our solar system moved the way they did. Yet, through careful study of observational data, they were able to formulate quantitative relations describing planetary motion. Notable examples include Kepler's laws, and even Isaac Newton's law of gravity (Kutner, 2003, page 434). It was only much later, after Albert Einstein published his theory of general relativity, that the physics underlying the aforementioned laws became understood by physicists.

As scientists developed a better understanding of physics, purely data-driven modelling lost traction in favour of physics-based modelling. Instead of using data to continuously develop new models from scratch, it was far more efficient for scientists to interpret data within pre-existing physical models in an effort to improve these models through modifications and extensions. Additionally, PBM offers explanations of *why* things are the way they are, while traditional DDM only explains *how* they are. For both research scientists and engineers (especially in high-stakes applications), this is a major disadvantage of DDM. Thus, for DDM to be worthwhile, it must offer something which PBM cannot, such as superior computational efficiency or high modelling accuracy for phenomena that are poorly understood and/or difficult to model using PBM.

[11] We use the LAPACK routine "?gesv" accessed through the SciPy library in our numerical experiments on 2D heat transfer problems.

As our understanding of both physics and numerical methods is ever-increasing, this is a tall order for DDM to match. However, as a growing number of applications migrate to big-data regimes, DDM is becoming increasingly relevant. Furthermore, as discussed in Chapter 1, the advent of modern computing infrastructure, combined with new, powerful data-driven techniques, has levelled the playing field between PBM and DDM. Deep neural networks (DNNs) have been particularly important in this development. We will discuss the basics of DNNs at the end of this chapter, in Section 2.3.3. For now, we will continue our high-level presentation of DDM.

In our study of the literature, we have identified two main approaches to DDM: Equation discovery and time series forecasting. The most important concepts of these two approaches will be presented below, along with some examples of their applications. Finally, this section will be concluded by a comparison of the two approaches.

Equation Discovery

Equation discovery is data-driven modelling in the spirit of the early experimental physicists; its main objective is to discover an *explicit* governing equation from observational data. Once a governing equation has been discovered, predictions for presently unseen scenarios can be made by solving the discovered equation. While the earliest forms of equation discovery were limited in their power due to manual data analysis, digital technologies have recently enabled the discovery of far more complex dynamics than was previously possible.

An interesting work on equation discovery is that by Cranmer et al. (2020). In that work, the authors first train a graph neural network (GNN) to capture the dynamics of the system at hand, before using symbolic regression to recover an explicit governing equation describing the dynamics learnt by the GNN. They demonstrate their approach by recovering the governing equations of known systems such as damped oscillators, and they also discover a novel equation describing dark matter overdensity from state-of-the-art dark matter simulations. In other works on equation discovery, such as those by Raissi et al. (2018) and Brunton et al. (2016), some basic structure is imposed on the unknown governing equation for mathematical convenience.[12] Both these works assume that the quantity of interest, which we denote u, is governed by an equation on the form

$$\frac{\mathrm{d}u}{\mathrm{d}t} = f(u), \tag{2.22}$$

where f is an unknown function that must be learnt from observations. Brunton et al. (2016) use sparse regression techniques to learn f, while Raissi et al. (2018) learn f using a simple neural network. In both works, the learnt functions f successfully capture the behaviour of a variety of physical systems, including the chaotic Lorenz equations. Equation discovery in this form is also studied by Ayed et al. (2019), who consider the special case when the state of the system can only be partially observed.

Time Series Forecasting

In contrast to equation discovery, time series forecasting does not aim to learn an explicit representation of a system's dynamics. Instead, a hidden representation of the dynamics

[12]It is important to note that these works are still distinctly different to works in parameter discovery, which we listed in Chapter 1 as one of the main approaches to HAM. In parameter discovery problems, the functional form of the system's governing equation is pre-imposed based on physical knowledge. Here, the imposed mathematical form is still highly general and does not stem from knowledge of system-specific physics.

is learnt and parametrized by a data-driven model – typically a neural network. This model is then used to make predictions without ever producing an explicit governing equation. Typically, the model is designed to capture the dynamics of the system in the form of a so-called transition function. For a general state variable u, the transition function maps the value of u at some time level n to its value at the subsequent time level $n+1$ (possibly by utilizing historic values of u in addition to its current value). The role of the transition function, which we denote τ, can be expressed mathematically as

$$u(t^{n+1}) = \tau(u(t^n), \ldots, u(t^{n-n_{\text{hist}}})), \tag{2.23}$$

where n_{hist} denotes the number of historic data points needed to predict u at the new time level.[13] The goal of learning an efficient parametrization of τ can be achieved e.g. through the use of traditional regression techniques or more recent techniques based on neural networks. Over the last decades, several research papers (Cai et al., 2019; Hill et al., 1996; Jain and Kumar, 2007; Pala and Atici, 2019; Zhang, 2003) have advocated the use of neural networks for time series forecasting, either as complete models or for use in combination with traditional regression techniques. In the rest of this section, we therefore focus on NN-based time series forecasting.

A wide variety of different neural network types can be used to parametrize the time series transition function, as discussed in the well-organized review of DNN-based time series forecasting by Mahmoud and Mohammed (2021). While some early research advocated the use of simple, fully connected neural networks (Hill et al., 1996; Kaastra and Boyd, 1996), recurrent neural networks (RNNs) and variations thereof have recently been dominating the field (Bai et al., 2018). A particularly popular variation of RNNs is long short-term memory networks (LSTMs), which have been used successfully for e.g. wind forecasting (Chen et al., 2018), solar activity forecasting (Pala and Atici, 2019), (short-term) precipitation forecasting (Shi et al., 2015), and COVID-19 infections forecasting (Zeroual et al., 2020). Zeroual et al. (2020) additionally consider variational autoencoders (VAEs) in their COVID-19 forecasting study. VAEs are also considered by Bao et al. (2017), who use them in combination with LSTMs and wavelet transforms to predict stock prices. Yet another alternative approach is to use the recently proposed temporal convolutional networks, which are demonstrated by Bai et al. (2018) and Wan et al. (2019) for univariate and multivariate applications, respectively. In conclusion, the examples above illustrate that a wide array of powerful DNN-based techniques is available for time series forecasting.

A Comparison of Equation Discovery and Time Series Forecasting

In comparison to equation discovery, time series forecasting has the benefit of being self-contained, in the sense that it does not rely on analytical or numerical solution methods for producing predictions. This is especially beneficial when the true governing equation cannot be solved analytically. In such cases, the accuracy of DDM based on equation discovery will be limited by the finite accuracy of the numerical method used to solve the governing equation. Since DDM based on time series forecasting does not rely on any numerical solver, its accuracy is only limited by the extent to which the model has learnt the underlying physics. Thus, a model based on time series forecasting will provide predictions with zero error if the underlying physics is learnt perfectly. Consequently, time series forecasting has a possible edge over equation discovery in terms of accuracy.

[13]For example, we will have $n_{\text{hist}} = 0$ for any system with the Markov property (future behaviour depends only on current state), while we will have $n_{\text{hist}} > 0$ e.g. for systems experiencing hysteresis.

However, when it comes to interpretability and trustworthiness, equation discovery is the superior approach. Since equation discovery provides an explicit representation of the learnt dynamics, the knowledge instilled in the DDM can be analyzed using both physics-based and mathematical frameworks. This enables *a posteriori* verification of the learnt dynamics as well as the development of stability conditions and error bounds. Such analyses are not facilitated in the same way by DDMs based on time series forecasting, since no explicit representation of the learnt dynamics is provided. Thus, equation discovery and time series prediction each have their individual strong and weak points. Which approach to use for any given problem must therefore be considered on a case-by-case basis.

2.3.2. Data-Driven Modelling for Heat Transfer Problems

In this section, we will describe and justify the DDM for unsteady heat transfer modelling that we use in our numerical experiments (cf. Chapter 4). To this end, we first have to decide whether the model should be based on equation discovery or time-series forecasting. We therefore begin by discussing this decision. Further details of the chosen model will be covered thereafter – first in 1D, then in 2D.

Both equation discovery and time series prediction are generally well-suited approaches to data-driven modelling of unsteady heat transfer. However, for the purposes of the present work, time series prediction is the most relevant approach. The most important reason for this is that we want to study cases where the heat equation cannot be solved analytically. As discussed above, equation discovery-based models are then reliant upon numerical methods like the Implicit Euler FVM to provide predictions. Using a numerical method is undesirable in two ways: 1) it increases model complexity, which makes analysis of the model and its predictions more difficult, and 2) it introduces numerical error, thereby providing an upper bound to the model's predictive accuracy even when the full physics is learnt perfectly. This would put DDM at an unfair disadvantage in our numerical experiments (cf. Chapter 4), where predictive accuracy is of great concern. We therefore use DDM based on time series forecasting for the experiments in this work.

On the basis of the research on time series forecasting cited above, we choose to use a deep neural network to learn a representation of the transition function τ. To have a well-defined learning problem for the DNN, we must also choose the number n_{hist} of historic data points to include as input to the DNN, in addition to the data from the present time level.[14] As any smooth function T can be a solution of the heat equation for some choice of P, there exists no upper bound on n_{hist} which guarantees that $T(t^{n+1})$ can be uniquely identified given $\{T(t^n), T(t^{n-1}), \ldots, T(t^{n-n_{\text{hist}}})\}$. Furthermore, there exists no general bound on n_{hist} which holds in a certain percent of all possible scenarios. Instead, an appropriate value of n_{hist} must be chosen based on the particular application at hand. We therefore make the arbitrary assumption that $n_{\text{hist}} = 0$, and make sure that we define our numerical experiments such that this does not result in an ill-posed DNN learning objective. In other words, we assume that

$$\boldsymbol{T}^{n+1} = \tau(\boldsymbol{T}^n). \tag{2.24}$$

yields a well-defined transition function τ, and we want to train a DNN to approximate

[14]These considerations apply to so-called feed-forward NNs, which is the kind of DNNs used in the present work. For e.g. recurrent NNs, the handling of historic data would be taken care of automatically by the NN.

this function. For one-dimensional problems, the ideal DNN mapping is then defined by

$$\begin{aligned} \mathrm{DNN}_{\mathrm{T}} : \mathbb{R}^{N_j+2} &\to \mathbb{R}^{N_j} \quad \text{such that } \boldsymbol{T}_{\mathrm{d}}^{n+1} = \boldsymbol{T}_{\mathrm{ref}}^{n+1}. \\ \boldsymbol{T}_{\mathrm{d}}^{n} &\mapsto \boldsymbol{T}_{\mathrm{d}}^{n+1} \end{aligned} \tag{2.25}$$

Here, the subscript $_{\mathrm{d}}$ is used to denote DDM-predicted temperature profiles, while the subscript $_{\mathrm{ref}}$ is used to denote observed reference data, i.e. solutions of the true governing equation sampled at distinct spatial and temporal locations. For simplicity, we use the same spatial and temporal discretization here as in Section 2.2.3. $\mathrm{DNN}_{\mathrm{T}}$ is here taken to be the ideal mapping that the DNN will be trained to approximate, but we will later also use the same notation to refer to DNNs trained to approximate this mapping; the assumed meaning will always be clear from context. As we did for the PBM discussed in the previous section, we assume that the boundary condition of our system are known, and this is the reason why the DNN output has two fewer components than the DNN input. As the BCs are known, we do not need to predict them, so the DNN output does not have any component describing the boundary temperature. However, we do want to include the boundary temperatures in the DNN input, as they contain useful information about the system at hand. Note also that, if $\boldsymbol{T}_{\mathrm{d}}^{0} = \boldsymbol{T}_{\mathrm{ref}}^{0}$, then the ideal mapping guarantees $\boldsymbol{T}_{\mathrm{d}}^{n} = \boldsymbol{T}_{\mathrm{ref}}^{n}$ also for all $n > 0$, as desired. Thus, if the DNN successfully learns the ideal mapping, then the DDM-generated time series $\{\boldsymbol{T}_{\mathrm{d}}^{n}\}_{n=0}^{N_t-1}$ will be precisely equal to the true time series $\{\boldsymbol{T}_{\mathrm{ref}}^{n}\}_{n=0}^{N_t-1}$.

For two-dimensional problems, the ideal DNN mapping is defined analogously to that of the 1D case described above:

$$\begin{aligned} \mathrm{DNN}_{\mathrm{T}} : \mathbb{R}^{(N_j+2)(N_i+2)} &\to \mathbb{R}^{N_j N_i} \quad \text{such that } \boldsymbol{T}_{\mathrm{d}}^{n+1} = \boldsymbol{T}_{\mathrm{ref}}^{n+1}. \\ \boldsymbol{T}_{\mathrm{d}}^{n} &\mapsto \boldsymbol{T}_{\mathrm{d}}^{n+1} \end{aligned} \tag{2.26}$$

The only difference between this definition and the 1D definition is that we have increased the dimensionality of the DNN's input space and output space to accommodate the extra spatial dimension. The discrete temperature vectors are defined using the same 2D discretization as was used in Section 2.2.3.

As the brief literature review given earlier in this chapter has shown, the mapping DNN_{T} can be learnt by a multitude of different types of DNNs using a score of different DNN training routines. Specific details regarding the DNN architecture and training routines used in our numerical experiments are given in Section 4.1. However, as promised, the basics of deep neural networks will be covered briefly in the next section.

2.3.3. An Introduction to Deep Neural Networks

In this section, we briefly cover some important concepts related to DNNs. We begin defining what we mean by the term DNN. Subsequently, we explain how DNNs can learn mappings like those defined in Equations (2.25) and (2.26). Along the way, we will also introduce some DNN terminology which will be used throughout the rest of this thesis. We highlight the use of the excellent textbooks by Nielsen (2015) and Goodfellow et al. (2016) as source material for this section. Since these works are cited here, they will not be cited again later in the section. However, other source material will be cited in the text wherever relevant.

Before we discuss neural networks specifically, let us say some words about machine learning at large. Machine learning is a branch of computer science concerned with the study of models that learn from experience. To be precise about what we mean by learning, we cite the definition given by Mitchell (1997):

Definition A computer program is said to *learn* from experience $\mathbb{E}$ with respect to some class of tasks $\mathbb{T}$ and performance measure $\mathbb{P}$, if its performance at tasks in $\mathbb{T}$, as measured by $\mathbb{P}$, improves with experience $\mathbb{E}$.

For example, $\mathbb{T}$ can be to predict the future temperature of a system given its current and past temperatures. $\mathbb{E}$ is then the feedback that the program has received on its earlier attempts at making such predictions, while $\mathbb{P}$ could e.g. be the absolute value of the difference between the predicted future temperature and the true future temperature.

A *neural network* (NN) is a special kind of machine learning model – i.e. a model capable of learning. In fact, the Universal Approximation Theorem guarantees that a neural network can learn *any* function, no matter how complicated. The name "neural network" alludes to the structure of these models, which is inspired by human brains. Just like brains, NNs consist of a (large) number of individual interconnected processing units, which we call *neurons*. Each neuron is associated with a set of weights $\boldsymbol{w}$ and (possibly) a bias β. A neuron uses $\boldsymbol{w}$ and β, which are collectively known as its *parameters*, to compute its output when given some input. Different types of neurons process their input differently. Additionally, different types of NNs organize their neurons differently. However, one often finds that NNs organize their neurons in *layers*, which are groups of neurons that operate (independently of each other) to produce some collective output when given a shared input. Again, different types of layers utilize their neurons differently to compute its output. Although it is beyond the scope of this section to consider all the possibilities, we mention fully connected layers (which we will return to later), convolutional layers and recurrent layers as important examples. We refer to the specific combination of layers used to construct an NN as the NN's *architecture*. Furthermore, we say that a neural network is *deep* if it consists of many layers. However, there exists no conventional lower bound for what counts as "many" in this context. For instance, an NN might be considered deep if it consists of more than one so-called hidden layers. A hidden layer is a term used to describe any layer which is not the NN's input layer (the layer which receives the NN's input) or its output layer (the layer responsible for producing the NN's final output). The appropriate number of layers for any given application must be determined by the NN developer on a case-by-case basis. This also applies to the number of neurons in each layer. We typically use the term *hyperparameters* when referring to model parameters that must be chosen by the developer and whose optimal values are not determined by the model itself. In addition to layer size and the number of layers, important hyperparameters include e.g. batch size and learning rate, both of which we will encounter later in this section.

For the numerical experiments described in Chapter 4, we use DNNs with fully connected layers. We will therefore have a brief look at how these layers work. In a fully connected layer, all neurons operate on the layer's full input, which is a vector we denote $\boldsymbol{\eta}$. The neuron's output z, which is a scalar, is then computed as $z = \boldsymbol{w} \cdot \boldsymbol{\eta} + \beta$ (cf. Figure 2.2a). Subsequently, the individual neuron outputs are collected in a vector $\boldsymbol{z}$, which is the output of the layer as a whole. The output $\boldsymbol{z}$ is then passed through a non-linear *activation function* Φ to produce $\boldsymbol{\phi} = \Phi(\boldsymbol{z})$. The activated output vector $\boldsymbol{\phi}$ is finally used as the input of the subsequent fully connected layer. Thus, the input of all neurons in a fully connected layer depends on the output of all neurons in the preceding layer, as illustrated in Figure 2.2b.

Let us take a moment to consider the activation function Φ in greater detail. First of all, the non-linearity of the activation function is essential, because without it, an NN with fully connected layers would just be a composition of matrix multiplications.

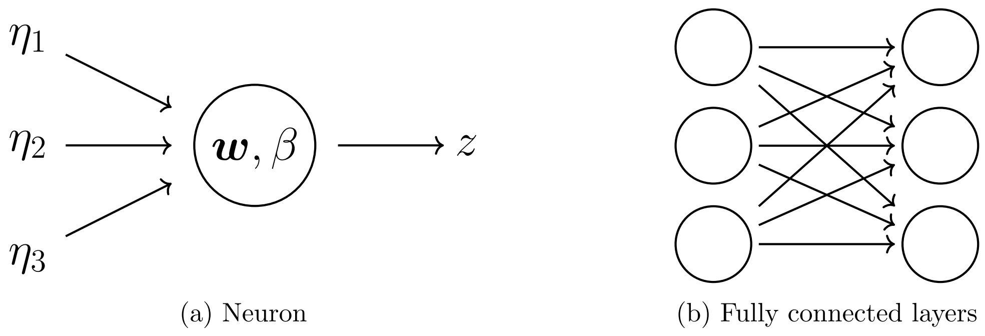


Figure 2.2.: Left: An illustration of a neuron receiving an input vector $\boldsymbol{\eta} = [\eta_1, \eta_2, \eta_3]$ and outputting a scalar $z = \boldsymbol{w} \cdot \boldsymbol{\eta} + \beta = w_1\eta_1 + w_2\eta_2 + w_3\eta_3 + \beta$. $\boldsymbol{w} = [x_1, w_2, w_3]$ are the neuron's weights, and β is its bias.
Right: An illustration of the connections between two fully connected layers, each consisting of three neurons.

Secondly, any non-linear function is (in principle) fit for use as an activation function in an NN. However, certain choices are more popular than others. Early works on neural networks often used the so-called sigmoid function $1/(1 + \exp(-z))$, but this function suffers from a phenomenon known as *saturation*. We say that an activation function Φ is saturated if a large change in z results in a comparatively small change in $\Phi(z)$. This is undesirable, because it inhibits information flow through the network due to phenomena known as exploding and vanishing gradients. A more modern choice of activation function is the Rectified Linear Unit (ReLU), which is defined as

$$\mathrm{ReLU}(z) = \begin{cases} z & z \geq 0, \\ 0 & z < 0. \end{cases} \tag{2.27}$$

ReLU does not suffer from exploding/vanishing gradients in the same way as the sigmoid function (Xu et al., 2015). However, the fact that it completely disregards all negative z may also inhibit learning. This issue is addressed by the LeakyReLU activation function (Maas et al., 2013). It is defined as

$$\mathrm{LeakyReLU}(z) = \begin{cases} z & z \geq 0, \\ -\Lambda z & z < 0, \end{cases} \tag{2.28}$$

where Λ is a constant chosen by the NN developer.[15]

So far, we have talked a lot about what a (deep) neural network *is*, but not so much about how it actually *learns*. On the most fundamental level, an NN learns by updating the parameters of its neurons such as to obtain better performance, as measured by the performance measure $\mathbb{P}$. This learning process is commonly referred to as NN *training*, and it can be carried out in a multitude of different ways. Here, we will focus on so-called *supervised learning*. Supervised learning requires the preparation of a number of *data examples*, each consisting of an NN input and a corresponding target output. For any data example, the NN takes the input and produces an output which ideally should be equal to the target output. The closeness of the NN output to the target output is measured by a *cost function*. Generally, the cost function is defined to output small values if the NN output is close to the target output and comparatively large values when the output is dissimilar to the target output. Thus, the NN's learning objective –

[15]The original incarnation of LeakyReLU used $\Lambda = 0.01$ (Maas et al., 2013).

which is to make its output similar to the target output – is equivalent to minimizing the cost function.

To minimize the cost function, NNs make use of an algorithm known as *backpropagation*, which is essentially a clever formulation of the chain rule for partial derivatives.[16] Using the backpropagation algorithm, it is possible to calculate the partial derivative of the cost function with respect to any of the individual parameters of the NN's neurons. With these gradients it is possible to optimize the parameters such as to minimize the cost function for all data examples available for training. Exactly how this is done depends on the optimization algorithm (also know as the *optimizer*) that is used.

For training NNs, optimizers based on stochastic gradient descent have proven to be highly effective in practice. These optimizers divide the full set of training examples into randomly chosen (small) groups known as *batches*. For each batch, the average cost of all the examples in the batch is computed. The NN's parameters are then updated in the direction of negative gradient of the batch-averaged cost. We refer to the process of updating the NN's parameters given the gradients from a single batch as a *training iteration*. The size of the updates naturally depends on the size of the gradients, but it also depends on a model parameter known as the *learning rate*. Simply put, a larger learning rate also yields larger updates per training iteration. For advanced optimizers, other factors are also taken into account when determining the size of the updates. For example, the Adam optimizer (Kingma and Ba, 2014) used in our numerical experiments (cf. Chapter 4) accounts for two further observations: 1) If the gradients have been pointing "in the same direction" for many subsequent training iterations, it might increase efficiency to make larger updates in that direction until the gradients are observed to change direction. 2) Some parameters may benefit from larger updates than others, depending e.g. on the parameters' relative importance and have often they contribute significantly to the final output of the NN. An excellent overview of commonly used optimizers (including the Adam optimizer) is given by Ruder (2016).

We conclude the present section by briefly covering the concepts of validation and evaluation (also known as testing) in the context of NNs. Validation and evaluation share a common objective, namely to assess the performance of an NN. The difference between the two is that validation refers to *pre*-deployment assessment, while evaluation refers to *post*-deployment assessment. Validation also has a clear analogy to training, which evaluation does not: Training is performed such that the optimizer can adjust the parameters of the neurons, while validation is performed such that the model developer can adjust the hyperparameters of the NN as a whole. In contrast, no changes are to be made on the basis of the evaluation results. The evaluation results constitute the final measure of the NN's performance *after* its development has been completed.

[16]Nielsen (2015, chapter 2) provides a particularly pedagogical explanation of backpropagation.

3. The Corrective Source Term Approach

In this chapter, we formally introduce and justify the Corrective Source Term Approach (CoSTA) to hybrid analysis and modelling (HAM). We define CoSTA to encompass any hybrid model developed by perturbing the governing equation of a physics-based model with a corrective source term generated using data-driven techniques. The general rationale behind perturbing a PBM with a corrective source term is presented in Section 3.1. Subsequently, in Section 3.2, we consider CoSTA's underlying theoretical framework in the context of heat transfer problems. Lastly, Section 3.3 is dedicated to a discussion on how and why we propose to use data-driven techniques (and deep neural networks, in particular) for generating the corrective source term.

3.1. Theoretical Justification of CoSTA

As discussed in Section 2.2, physics-based modelling involves formulating and solving a governing equation for the system to be modelled. In this section, we consider a general governing equation describing some quantity u on a domain Ω with boundary $\partial\Omega$. Our goal is to show that a cleverly defined corrective source term can be used to correct *any* error in a physics-based model of u. To this end, let us assume that u is the exact solution of a governing equation on the following general form (including BCs):

$$\mathcal{N}_{\Omega} u = f \text{ in } \quad \Omega, \tag{3.1}$$

$$\mathcal{N}_{\partial\Omega} u = g \text{ on } \partial\Omega. \tag{3.2}$$

Here, $\mathcal{N}_{\Omega}$ and $\mathcal{N}_{\partial\Omega}$ represent general, possibly non-linear operators[1] acting on u, while f and g are general functions. Note that this formulation also captures scenarios where there are multiple governing equations. In such scenarios, u is a vector, and f and g are vector-valued functions.

Assume now that we have a PBM designed to predict u, and let $\tilde{u}$ denote the PBM's prediction of the true solution u. We argue that if $\tilde{u} \neq u$, then the error in PBM can always be corrected using a corrective source term. The argument begins by observing that any error in the PBM must stem from one of the following sources:

1. The true function f in Equation (3.1) is unknown, so it is approximated by $\tilde{f}$.
2. The true operator $\mathcal{N}_{\Omega}$ in Equation (3.1) is unknown, so it is approximated by $\widetilde{\mathcal{N}}_{\Omega}$.
3. The true function g in Equation (3.2) is unknown, so it is approximated by $\tilde{g}$.
4. The true operator $\mathcal{N}_{\partial\Omega}$ in Equation (3.2) is unknown, so it is approximated by $\widetilde{\mathcal{N}}_{\partial\Omega}$.
5. A combination of the above.

[1]For u to be uniquely defined, $\mathcal{N}_{\partial\Omega}$ must be the unity mapping for at least one location on $\partial\Omega$.

6. We know the true governing equation (3.1) and the true boundary conditions (3.2), but we are unable to solve these equations exactly. To obtain a prediction $\tilde{u}$, we must therefore solve some approximation of the true system constituted by Equations (3.1) and (3.2). This puts us in one of the cases described above. One typical example is to approximate $\mathcal{N}_\Omega$ using a numerical operator $\mathcal{N}_{\text{num}}$ e.g. based on finite-difference approximations, which puts us in Case 2.

As stated above, Case 6 is always mathematically equivalent to one of the other cases. Additionally, Cases 3 and 4 are analogous to Cases 1 and 2 because $\mathcal{N}_{\partial\Omega}$ and g play exactly the same roles in Equation (3.2) as $\mathcal{N}_\Omega$ and f do in Equation (3.1); if any error in Equation (3.1) can be corrected by adding a cleverly defined source term to its right-hand side, then adding an analogously defined source term to the right-hand side of Equation (3.2) must necessarily correct any error in the latter equation. To show that it is always possible to correct any error in the PBM through the use of a corrective source term, it therefore suffices to consider Cases 1 and 2, and combinations thereof.

Suppose now that the PBM-predicted solution $\tilde{u}$ is given as the solution of the following governing equation:

$$\widetilde{\mathcal{N}}_\Omega \tilde{u} = \tilde{f} \text{ in } \quad \Omega, \tag{3.3}$$

$$\mathcal{N}_{\partial\Omega} \tilde{u} = g \text{ on } \partial\Omega. \tag{3.4}$$

This formulation encompasses both Case 1 (for $\widetilde{\mathcal{N}}_\Omega = \mathcal{N}_\Omega$ and $\tilde{f} \neq f$), Case 2 (for $\widetilde{\mathcal{N}}_\Omega \neq \mathcal{N}_\Omega$ and $\tilde{f} = f$), and combinations thereof (for $\widetilde{\mathcal{N}}_\Omega \neq \mathcal{N}_\Omega$ and $\tilde{f} \neq f$). Furthermore, suppose we modify the system above by adding a source term $\hat{\sigma}$ to Equation (3.3), and let the solution of the modified system be denoted $\hat{\tilde{u}}$. Then, the modified system reads

$$\widetilde{\mathcal{N}}_\Omega \hat{\tilde{u}} = \tilde{f} + \hat{\sigma} \text{ in } \quad \Omega, \tag{3.5}$$

$$\mathcal{N}_{\partial\Omega} \hat{\tilde{u}} = g \qquad \text{on } \partial\Omega. \tag{3.6}$$

and the following theorem holds.

Theorem Let $\hat{\tilde{u}}$ be a solution of Equations (3.5) and (3.6), and let u be a solution of Equations (3.1) and (3.2). Then, for all operators $\widetilde{\mathcal{N}}_\Omega$, $\mathcal{N}_\Omega$, $\widetilde{\mathcal{N}}_{\partial\Omega}$ and $\mathcal{N}_{\partial\Omega}$ and all functions f, $\tilde{f}$, g and $\tilde{g}$ such that $\hat{\tilde{u}}$ and u are uniquely defined, there exists a function $\hat{\sigma}$ such that $\hat{\tilde{u}} = u$.

Proof: Define the residual r of the PBM's governing equation (3.3) as[2]

$$r = \widetilde{\mathcal{N}}_\Omega u - \tilde{f}. \tag{3.7}$$

If we set $\hat{\sigma} = r$ in Equation (3.5), we then obtain

$$\begin{aligned} \widetilde{\mathcal{N}}_\Omega \hat{\tilde{u}} &= \tilde{f} + \hat{\sigma} \\ &= \tilde{f} + \widetilde{\mathcal{N}}_\Omega u - \tilde{f} \\ &= \widetilde{\mathcal{N}}_\Omega u \\ \implies \quad \hat{\tilde{u}} &= u \qquad \blacksquare \end{aligned}$$

[2]Note that our definition is in some sense opposite of common practice; we have defined the residual by inserting the true solution into the approximate equation rather than inserting the approximate solution into the true equation. The latter is the conventional approach, and is used e.g. in truncation error analysis (LeVeque, 2002, chapter 8). The reason for our choice is two-fold: 1) It yields the simplest proof of the theorem. 2) When observing a real-world system, it is often easier to measure its state than to find the exact governing equation describing said state.

The theorem above proves that, for any error in the PBM's governing equation (3.3), there always exists a source term $\hat{\sigma}$ which we can add to that equation such that the solution $\hat{\tilde{u}}$ of the modified governing equation (3.5) is equal to the true solution u. If the source of the error in $\tilde{u}$ was instead found in Equation (3.4), which defines the PBM's boundary conditions, we would modify that equation analogously. That is, we would add a corrective source term to its right-hand side, and the source term would be given by the residual of Equation (3.4) – not Equation (3.3) – with the true solution u inserted in the place of $\tilde{u}$. Thus, the true solution u of the true governing equations can always be retained by modifying an erroneous PBM with a corrective source term. This observation is the principal theoretical justification of the Corrective Source Term Approach presented in this work.

From a practical point of view, some challenges related to the definition of the corrective source term still remain. The residual r defined in Equation (3.7), depends on the true solution u. Thus, setting $\hat{\sigma} = r$ in the modified PBM means that the prediction $\hat{\tilde{u}}$ depends on u. For *a posteriori* analyses where u is known, this does not pose any issues. However, it is an issue if we want to make *a priori* predictions of u, because u itself is then unknown. When u is unknown, we cannot explicitly compute the residual r (as defined in Equation (3.7)) and insert it as a source term in Equation (3.5). We discuss how to circumvent this issue in Section 3.3.

3.2. Applying CoSTA to Heat Transfer Problems

In this section, we provide an in-depth discussion of how the general CoSTA framework presented in Section 3.1 can be applied in practice. More specifically, we will apply the framework to the heat equation and the Implicit Euler FVM for unsteady heat transfer problems, which we presented in Section 2.2.

Let us begin by considering the heat equation. This equation is the true governing equation for unsteady heat transfer, and is thus the manifestation of Equation (3.1) for unsteady heat transfer problems. We have written the heat equation on both integral and differential forms in various dimensions, and all these forms can be reconciled with the general framework of Section 3.1. As an example, we demonstrate how to do this for the 1D heat equation on differential form (2.10) with Dirichlet BCs. For any of the other forms of the heat equation, the process would be analogous to the one used below.

Our quantity of interest in this example is the temperature T, so T now plays the role of the general variable u used in Section 3.1. Furthermore, through a rearrangement of terms, Equation (2.10) can be rewritten as

$$\rho c_V \frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(kA\frac{\partial T}{\partial x}\right) = P. \tag{3.8}$$

Comparing Equation (3.8) to Equation (3.1), it is clear that we have

$$u = T, \quad \mathcal{N}_\Omega = \rho c_V \frac{\partial}{\partial t} - \frac{\partial}{\partial x}\left(kA\frac{\partial}{\partial x}\right) \quad \text{and} \quad f = P. \tag{3.9}$$

As for the boundary conditions, we know from the definition of Dirichlet BCs that the temperature T is always equal to some function $T_{\partial\Omega}$ at all boundary locations. In Equation (3.2), we then have that $\mathcal{N}_{\partial\Omega}$ is the unity operator and $g = T_{\partial\Omega}$ is the prescribed boundary temperature.

Continuing the example above, suppose that we do not know the true source term P or the true conductivity k, such that these must be approximated by $\widetilde{P}$ and $\tilde{k}$, respectively.

The true governing equation (Equation (3.8) in this example) is then approximated by

$$\rho c_V \frac{\partial T}{\partial t} - \frac{\partial}{\partial x}\left(\tilde{k} A \frac{\partial T}{\partial x}\right) = \widetilde{P}, \tag{3.10}$$

which is analogous to Equation (3.3). Consequently, we observe that the approximate operator $\widetilde{\mathcal{N}}_\Omega$ and the approximate function $\tilde{f}$ are

$$\widetilde{\mathcal{N}}_\Omega = \rho c_V \frac{\partial}{\partial t} - \frac{\partial}{\partial x}\left(\tilde{k} A \frac{\partial}{\partial x}\right) \quad \text{and} \quad \tilde{f} = \widetilde{P} \tag{3.11}$$

in the present example. Using the relations above, we could also define the corrective source term $\hat{\sigma}$. In a PBM where the governing equation can be solved exactly, this would indeed be the natural next step. However, when a numerical solver is required for solving the governing equation – as is generally the case for the heat equation – it is desirable to correct for the numerical error of the solver in addition to any modelling error in the governing equation. Defining the source term at this stage would enable us to capture only the latter error. We therefore move on to discussing how the Implicit Euler FVM fits in with the framework outlined in Section 3.1. The corrective source term will be defined subsequently as a correction to the FVM. This way, the corrective source term can capture both modelling error and discretization error.

In both one and two dimensions, the Implicit Euler FVM can be expressed on the form

$$\mathbb{A}\boldsymbol{T}_\mathrm{p}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}_\mathrm{p}^{n}\right), \tag{3.12}$$

where we now use the subscript $_\mathrm{p}$ to indicate that the solution of the system above is a PBM approximation of the true solution. Furthermore, we use the notation T_ref to refer to the true solution of the heat equation, and we let $\boldsymbol{T}_\mathrm{ref}$ denote T_ref evaluated at the interior nodes used to define the FVM expressed in Equation (3.12). Upon comparison of Equation (3.12) with Equation (3.3), we observe the following relations:

$$\widetilde{\mathcal{N}}_\Omega \leftrightarrow \mathbb{A}, \quad \tilde{u} \leftrightarrow \boldsymbol{T}_\mathrm{p}^{n+1}, \quad \tilde{f} \leftrightarrow \boldsymbol{b}, \quad u \leftrightarrow \boldsymbol{T}_\mathrm{ref}^{n+1}. \tag{3.13}$$

With these relations, we can define the corrective source term for the Implicit Euler FVM using the residual from Equation (3.7). The definition reads

$$\hat{\boldsymbol{\sigma}}^{n+1} = r = \mathbb{A}\boldsymbol{T}_\mathrm{ref}^{n+1} - \boldsymbol{b}\left(\boldsymbol{T}_\mathrm{ref}^{n}\right). \tag{3.14}$$

Note that we here write the corrective source term in bold because it is defined as a vector for the relations (3.13) given by the Implicit Euler FVM. We also add a time-level superscript to highlight the time-dependence of the corrective source term.

As in Section 3.1, we use the corrective source term to define a modified governing equation whose solution is exactly equal to the reference solution at all grid nodes and at all time levels. We use a subscript $_\mathrm{h}$ (for "hybrid analysis and modelling") to denote the solution of the modified system, and we write the modified system as

$$\mathbb{A}\boldsymbol{T}_\mathrm{h}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}_\mathrm{h}^{n}\right) + \hat{\boldsymbol{\sigma}}^{n+1}. \tag{3.15}$$

When testing our CoSTA-based HAM models in Chapter 4, this is the equation we solve to obtain the HAM predictions.

In Section 3.1, we explained how CoSTA can be used to correct errors in both the governing equation and in the prescribed BCs. However, with the definition of the corrective source term in Equation (3.14), it is not possible to correct the prescribed BCs because the discretized temperature vectors that appear in Equations (3.14) and (3.15) describe only the temperature at interior grid nodes. The boundary temperatures are therefore unaffected by $\hat{\boldsymbol{\sigma}}^{n+1}$ when it is defined as in Equation (3.14). In all our numerical experiments, we assume that the BCs are always known exactly, so the inability to correct the BCs is not a concern in the present work. However, if necessary, it is possible to reformulate the definition of the corrective source term in a way that permits correcting the BCs. We show how to do this in Appendix B.

3.3. Calculating the Corrective Source Term

In the previous two sections, we have established that it is generally possible to recover the true solution of any governing equation by modifying a PBM using a corrective source term. We have also seen how to define this corrective source term, both generally (in Equation (3.7)) and for the Implicit Euler FVM (in Equation (3.14)). As discussed in Section 3.1, these definitions can be used as is for *a posteriori* calculations where the quantity of interest (u in the general case, or $\boldsymbol{T}_{\mathrm{ref}}^{n+1}$ for the Implicit Euler FVM) is known. However, when the quantity of interest is unknown, i.e. when we want to compute the corrective source term *a priori* in order to make predictions into the future, these definitions cannot be used directly. In this section, we discuss how and why we propose to circumvent this issue using DNNs.

Co-dependence between unknown variables is actually quite common in real-world scenarios; consider for example the possibility of σ in Equation (2.12) being temperature-dependent. Then, in Equation (2.18), $\boldsymbol{T}^{n+1}$ depends on $\boldsymbol{\sigma}^{n+1}$ (through the vector $\boldsymbol{b}$) whilst $\boldsymbol{\sigma}^{n+1}$ also depends on $\boldsymbol{T}^{n+1}$ (through the constitutive relation defining σ). These situations can be resolved using iterative methods[3] where one first uses an approximation of $\boldsymbol{T}^{n+1}$ to calculate an approximation of $\boldsymbol{\sigma}^{n+1}$. The approximation of $\boldsymbol{\sigma}^{n+1}$ is then used to calculate a new approximation of $\boldsymbol{T}^{n+1}$ which is used to calculate a new approximation of $\boldsymbol{\sigma}^{n+1}$, and so on. The process continues until $\boldsymbol{\sigma}^{n+1}$ and $\boldsymbol{T}^{n+1}$ change comparatively little from one iteration to the next.

Unfortunately, for CoSTA, it is not possible to define an iterative scheme like the one above which converges to $\hat{\sigma} = r$ and $\hat{\hat{u}} = u$ as one would desire. This is because no matter which approximation of u we use to initiate the iterative process, the definition of the modified system (3.5) ensures that the corrected prediction $\hat{\hat{u}}$ will remain equal to the chosen initial approximation at all subsequent iteration levels. That is to say, the iterative process never leaves its starting point. This is illustrated below, where we attempt to obtain $\hat{\hat{u}} = u$ at some time level $n > 0$ using an iterative scheme with initial guess[4] $\hat{\hat{u}}^{n,0}$. Here, the second superscript denotes iteration level.

$$
\begin{aligned}
\text{Initial guess}: \quad & \hat{\hat{u}}^{n,0} \\
\text{First iteration}: \quad & \hat{\sigma}^{n,1} = \widetilde{\mathcal{N}}_{\Omega} \hat{\hat{u}}^{n,0} - \tilde{f}^{n} \\
\widetilde{\mathcal{N}}_{\Omega} \hat{\hat{u}}^{n,1} &= \tilde{f}^{n} + \hat{\sigma}^{n,1} \\
\widetilde{\mathcal{N}}_{\Omega} \hat{\hat{u}}^{n,1} &= \tilde{f}^{n} + \widetilde{\mathcal{N}}_{\Omega} \hat{\hat{u}}^{n,0} - \tilde{f}^{n} = \widetilde{\mathcal{N}}_{\Omega} \hat{\hat{u}}^{n,0} \\
\implies \hat{\hat{u}}^{n,1} &= \hat{\hat{u}}^{n,0}
\end{aligned}
$$

[3] Another option is to linearize σ, which means that σ is approximated by a function that is linear in T. However it is not clear how one would go about linearizing the corrective source term $\hat{\sigma}$ in terms of u in any meaningful way, so we do not discuss this option further.

[4] The initial guess is not to be confused with the initial condition u^0, which fixes u at the initial time level $n = 0$. The initial guess can e.g. be obtained by solving Equation (3.3).

It is clear that if we continue iterating as above, we find that $\hat{\hat{u}}^{n,i} = \hat{\hat{u}}^{n,0}$ for all $i \in \mathbb{N}$. Hence, $\hat{\hat{u}}^{n,i}$ cannot converge towards u^n as $i \to \infty$ unless we have $\hat{\hat{u}}^{n,0} = u^n$ by chance. We therefore need a different approach to determining the corrective source term, if we are to make use of CoSTA for *a priori* predictions.

As long as the governing equation of our process of interest is deterministic, its true solution u at any future time is, by definition, completely determined by its solution in the present and past. That is to say, for deterministic systems, it is in principle always possible to predict u at any future time given past and present data. It is then necessarily also possible to predict the residual defined in Equation (3.7). Our problem is that we do not know *how* to do this exactly, because we are not able to extract all the necessary information from the past and present data. This is where data-driven techniques enter the picture. As discussed in Section 2.3, data-driven techniques have displayed great capabilities of extracting and utilizing relevant information from large amounts of data. Indeed, this is what has motivated much of recent research into DDM. In DDM, one typically focuses on the quantity of interest itself, but here, we instead advocate the use of data-driven techniques for predicting the ideal corrective source term $\hat{\sigma} = r$. This way, we retain whatever useful information is already instilled in our PBM, which is a key point in the philosophy of HAM. All the data-driven techniques considered in Section 2.3 could conceivable be adapted to for the purpose of learning $\hat{\sigma}$. Thus, to determine which technique is optimal would be a comprehensive undertaking that is outside the scope of the present work. For the rest of this section, we will therefore focus mainly on explaining and justifying our chosen technique – deep neural networks.

Motivated by the fact that neural networks are universal function approximators and by the successes of deep learning as discussed in Sections 1.1 and 2.3, we choose to utilize a deep neural network (DNN) to predict the corrective source term. We then have to decide on what quantities to use as DNN input and as DNN target output during training. For the target output, we seemingly have two options: 1) use $\hat{\sigma}$ itself as target output, or 2) use u as target output, and insert the DNN prediction of u into Equation (3.7) to obtain the corrective source term. Because it facilitates re-purposing a pre-trained DNN from a purely data-driven model, the second option might seem appealing. However, this option yields a problem similar to the one we encountered in our discussion of iterative methods above; by construction, the solution of the modified governing equation (3.5) will be precisely the DNN's prediction of u. Essentially, our hybrid model then behaves exactly like a purely data-driven model followed by a computationally expensive unity operator. This is clearly unsatisfactory, so we have to go with the first option, i.e. training the DNN to predict $\hat{\sigma}$ directly.[5]

We now move on to presenting our choice of DNN input. Here, we draw inspiration from predictor-corrector methods used for numerical integration of ordinary differential equations. Predictor-corrector methods can be viewed as two-step algorithms consisting of a prediction step and a correction step. In the prediction step, one integration scheme is used to calculate a so-called "predictor", which is simply an approximation of the solution at the new time level given the solution at the old time level. In the corrector step, both the solution at the old time level and the predictor are used by a second integration scheme to compute the final prediction at the new time level. Suppose now that we have a discretized solution $\boldsymbol{T}_{\mathrm{h}}^{n}$ of the heat equation, and that we want

[5]Note that, in certain circumstances, it may be more efficient to predict certain components of $\hat{\sigma}$ rather than to predict $\hat{\sigma}$ as a whole. This is e.g. the case if $\hat{\sigma}$ is a sparse vector or matrix, such as in the example considered in Appendix A. However, this distinction is not relevant for the use-cases considered in the main text, so we will not discuss it further here.

to use CoSTA with the Implicit Euler FVM to predict $\boldsymbol{T}_{\mathrm{h}}^{n+1}$. We may then use the following procedure, which is analogous to the general predictor-corrector algorithm outlined above.

1. *Prediction step:* Compute the predictor $\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1}$ as the solution of the Implicit Euler FVM *without* the corrective source term, i.e. as the solution of

$$\mathbb{A}\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}_{\mathrm{h}}^{n}\right). \tag{3.16}$$

2. *Correction step:* Feed the predictor $\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1}$ as input to a DNN trained to predict $\hat{\boldsymbol{\sigma}}^{n+1}$, thereby obtaining the DNN-predicted corrective source term $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1}$. Then compute the corrected prediction $\boldsymbol{T}_{\mathrm{h}}^{n+1}$ as the solution of the Implicit Euler FVM *with* the corrective source term, i.e. as the solution of

$$\mathbb{A}\boldsymbol{T}_{\mathrm{h}}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}_{\mathrm{h}}^{n}\right) + \hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1}. \tag{3.17}$$

This procedure clearly suggests using $\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1}$ as DNN input, which is also essentially what we do in our experiments, except that we make one small modification: We have typically taken discrete temperature vectors like $\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1}$ to only contain components corresponding to interior grid nodes. However, we want the DNN to be able to utilize information from both the domain interior *and* the domain boundaries. For that reason, we extend $\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1}$ to also include the boundary temperatures at time level $n+1$ before passing it as input to the DNN. To avoid unnecessarily complex notation, we write $\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1}$ for temperature vectors both with and without boundary temperatures included; the implied meaning will generally be clear from context. With this clarification made, we can formally define the function we train the DNN to approximate. In the 1D case, the definition reads

$$\begin{aligned} \mathrm{DNN}_{\sigma} : \mathbb{R}^{N_j+2} &\to \mathbb{R}^{N_j} \quad \text{such that} \quad \hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1} = \hat{\boldsymbol{\sigma}}^{n+1}, \\ \widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1} &\mapsto \hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1} \end{aligned} \tag{3.18}$$

while the 2D definition is

$$\begin{aligned} \mathrm{DNN}_{\sigma} : \mathbb{R}^{(N_j+2)\cdot(N_i+2)} &\to \mathbb{R}^{N_j \cdot N_i} \quad \text{such that} \quad \hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1} = \hat{\boldsymbol{\sigma}}^{n+1}. \\ \widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1} &\mapsto \hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1} \end{aligned} \tag{3.19}$$

Later, we will additionally use the notation DNN_{σ} to refer to any DNN trained to approximate one of the functions defined in Equations (3.18) and (3.19); as with the temperature vectors, the implied meaning should always be clear from context. Note further that the dimensionality of the DNN's output is lower than the dimensionality of its input because the corrective source term, as defined in Equation (3.14), contains no components corresponding to grid nodes at the domain boundary. Note also that the condition $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1} = \hat{\boldsymbol{\sigma}}^{n+1}$ assumes that the DNN is trained perfectly, which is exceedingly difficult to achieve in practice. Thus, the condition will likely only be approximately satisfied by any real DNN. Our hypothesis is that even if the true corrective source term is only approximately represented by the DNN-prediction, using the DNN-prediction as a corrective source term will still yield better performance than comparable purely physics-based or purely data-driven modelling techniques. This hypothesis will be put to the test in Chapter 4.

We wrap up this section by summarizing the Corrective Source Term Approach (CoSTA), as proposed in the present chapter. In Section 3.1, we proved that for any physical system (Equations (3.1) and (3.2)) and any PBM approximating that system (Equations (3.3) and (3.4)), we can always define an augmented PBM (Equations (3.5) and (3.6)) which describes the behaviour of the physical system without error. More specifically, the PBM is augmented by a corrective source term (hence the name of the approach). This corrective source term is given as the residual of the PBM with the true solution inserted (cf. Equation (3.7)). Due to its dependence on the true solution, the corrective source term cannot be computed *a priori* using neither direct nor iterative methods. We therefore propose to use a deep neural network to approximate the corrective source term. For the modified Implicit Euler FVM (Equation (3.15)), we train the DNN to approximate the function DNN_σ as defined in Equation (3.18) (for 1D problems) or Equation (3.19) (for 2D problems). CoSTA does not impose any significant restrictions with regards to DNN architectures or training procedures, and it is outside the scope of the present work to determine an optimal architecture or an optimal training procedure. Our chosen DNN architecture and training procedure are described in Section 4.1, where we also discuss all other aspects of our experimental setup and methodology.

4. Numerical Experiments – Results and Discussions

In this chapter, we present a series of numerical experiments aimed at comparing the performance of stand-alone physics-based modelling (PBM), stand-alone data-driven modelling (DDM), and hybrid analysis and modelling (HAM) based on the corrective source term approach (CoSTA). The experiments consider time-dependent heat transfer problems in one and two dimensions. We mainly consider scenarios where modelling error is the dominant source of error in the PBM, because the benefit from data-driven techniques is greatest in such scenarios, as discussed in Chapter 1. However, to demonstrate the broad applicability of CoSTA-based HAM, we also include some experiments where discretization error is the only source of error in the PBM.

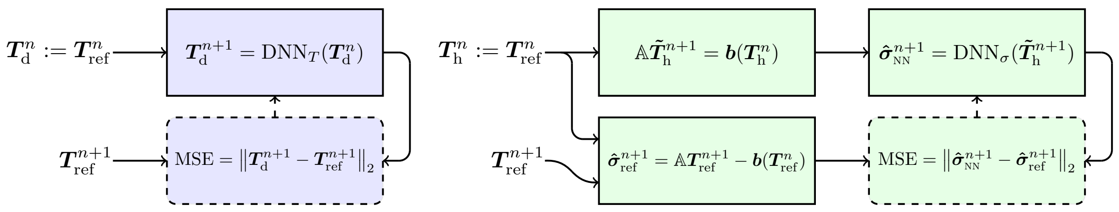
Our experimental setup and methodology is described in Section 4.1, while we present and discuss results from the experiments on one-dimensional and two-dimensional systems in Sections 4.2 and 4.3, respectively. In Section 4.4, we compare the uncertainty of predictions made using DDM and HAM, and, in Section 4.5, we conduct grid refinement studies to explore how the accuracy of PBM, DDM and HAM predictions relates to the spatial resolution at which predictions are made. Thereafter, in Section 4.6, we explore how the corrective source term of CoSTA can be interpreted in a physics context. The experiments and discussions of Sections 4.2, 4.3 and 4.5 are aimed at answering the research questions concerning accuracy and generalizability of PBM, DDM and HAM models, as formulated in Section 1.2, while Sections 4.4 and 4.6 address the research question concerning the trustworthiness of CoSTA-based HAM models. The content of this chapter is summarized and used to answer our research questions in Section 4.7.

4.1. Experimental Setup and Methodology

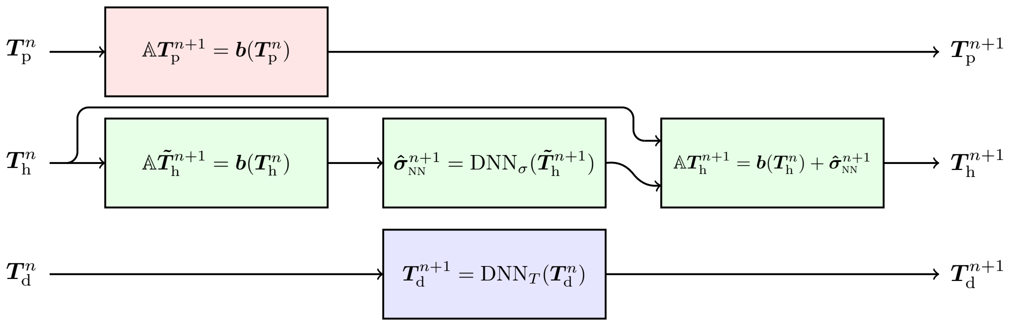
This section is dedicated to explaining the details of our experimental setup and methodology. All experiments considered in this chapter concern unsteady heat transfer in one and two dimensions. Thus, in all experiments, the true governing equation is the heat equation, defined e.g. in Equation (2.10) or Equation (2.11) for one- and two-dimensional problems respectively. For predicting the solutions of the heat equation, we use the PBM, DDM and HAM heat transfer models that were presented in Chapters 2 and 3. These are briefly summarized in Section 4.1.1. In Section 4.1.2, we move on to explain how we create reference data for the purposes of model evaluation and DNN training. Details on our DNN training procedure can be found in Section 4.1.3, while details concerning the DNNs themselves are given in Section 4.1.4. Finally, in Section 4.1.5, we cover the model evaluation procedure used to generate the experimental results presented herein.

4.1.1. Model Summary

Below, we summarize the PBM presented in Section 2.2, the DDM presented in Section 2.3 and the CoSTA-based HAM model presented in Sections 3.2 and 3.3. A visual representation of these models can be found in Figure 4.1.


$T_{\mathrm{d}}^{n} := T_{\mathrm{ref}}^{n}$
$T_{\mathrm{d}}^{n+1} = \mathrm{DNN}_T(T_{\mathrm{d}}^{n})$
T_{ref}^{n+1}
$\mathrm{MSE} = \|T_{\mathrm{d}}^{n+1} - T_{\mathrm{ref}}^{n+1}\|_2$
$T_{\mathrm{h}}^{n} := T_{\mathrm{ref}}^{n}$
$\mathbb{A}\tilde{T}_{\mathrm{h}}^{n+1} = b(T_{\mathrm{h}}^{n})$
$\hat{\sigma}_{\mathrm{NN}}^{n+1} = \mathrm{DNN}_\sigma(\tilde{T}_{\mathrm{h}}^{n+1})$
T_{ref}^{n+1}
$\hat{\sigma}_{\mathrm{ref}}^{n+1} = \mathbb{A}T_{\mathrm{ref}}^{n+1} - b(T_{\mathrm{ref}}^{n})$
$\mathrm{MSE} = \|\hat{\sigma}_{\mathrm{NN}}^{n+1} - \hat{\sigma}_{\mathrm{ref}}^{n+1}\|_2$

(a) Training procedures for the DNNs used in DDM (left) and HAM (right).


T_{p}^{n}
$\mathbb{A}T_{\mathrm{p}}^{n+1} = b(T_{\mathrm{p}}^{n})$
T_{p}^{n+1}
T_{h}^{n}
$\mathbb{A}\tilde{T}_{\mathrm{h}}^{n+1} = b(T_{\mathrm{h}}^{n})$
$\hat{\sigma}_{\mathrm{NN}}^{n+1} = \mathrm{DNN}_\sigma(\tilde{T}_{\mathrm{h}}^{n+1})$
$\mathbb{A}T_{\mathrm{h}}^{n+1} = b(T_{\mathrm{h}}^{n}) + \hat{\sigma}_{\mathrm{NN}}^{n+1}$
T_{h}^{n+1}
T_{d}^{n}
$T_{\mathrm{d}}^{n+1} = \mathrm{DNN}_T(T_{\mathrm{d}}^{n})$
T_{d}^{n+1}

(b) Time stepping procedures for PBM (top), HAM (middle) and DDM (bottom).

Figure 4.1.: Training and time stepping procedures for the three modeling approaches PBM (red), DDM (blue) and HAM (green). Note that PBM is not included in (a) because it does not require any training.

Physics-Based Modelling Our PBM of choice is the Implicit Euler FVM, which we defined in Section 2.2 for one- and two-dimensional problems. In both cases, it can be written on the form

$$\mathbb{A}\boldsymbol{T}_{\mathrm{p}}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}_{\mathrm{p}}^{n}\right), \tag{4.1}$$

where all quantities are defined as in Equation (2.18) (for 1D problems) or Equation (2.21) (for 2D problems). The subscript $_{\mathrm{p}}$ is used to indicate that the subscripted temperature vectors are PBM predictions.

Data-Driven Modelling To perform DDM, we use the DNN-based approach described in Section 2.3. That is to say, we train a deep neural network DNN_T to approximate the ideal mapping defined in Equation (2.25) (for 1D problems) or in Equation (2.26) (for 2D problems). The predictions of our DDM model, which we denote using a subscript $_{\mathrm{d}}$, are then given by

$$\boldsymbol{T}_{\mathrm{d}}^{n+1} = \mathrm{DNN}_T(\boldsymbol{T}_{\mathrm{d}}^{n}). \tag{4.2}$$

Hybrid Analysis and Modelling For our HAM model, we combine the Implicit Euler FVM from our PBM model with a DNN using CoSTA, as described in Sections 3.2 and 3.3. We use a subscript $_{\mathrm{h}}$ to denote HAM predictions, which we calculate using the following equation

$$\mathbb{A}\boldsymbol{T}_{\mathrm{h}}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}_{\mathrm{h}}^{n}\right) + \hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1}. \tag{4.3}$$

This equation is analogous to Equation (3.15), except we have replaced the ideal $\hat{\boldsymbol{\sigma}}^{n+1}$ with its DNN-approximation $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1}$, in accordance with the discussion in Section 3.3. The approximated corrective source term $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1}$ is defined by

$$\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1} = \mathrm{DNN}_\sigma(\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1}), \tag{4.4}$$

where DNN_σ is a DNN trained to approximate the ideal mapping defined in Equation (3.18) (for 1D problems) or Equation (3.19). Furthermore, as discussed in Section 3.3, the predictor $\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1}$ is defined by

$$\mathbb{A}\widetilde{\boldsymbol{T}}_{\mathrm{h}}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}_{\mathrm{h}}^{n}\right). \tag{4.5}$$

We highlight that, in any given experiment, we always use the same definitions of $\mathbb{A}$ and $\boldsymbol{b}$ in Equations (4.3) and (4.5) as in Equation (4.1).[1]

4.1.2. Data Generation

A central part of our data generation procedure is the method of manufactured solutions (MMS), which is a widely used method for verifying the correct implementation of numerical solvers of partial differential equations (PDEs) (Roache, 2002). The key concept of MMS is to first choose some conveniently expressible function, and then subsequently define the parameters of the PDE in question such that the chosen function satisfies the PDE. In our experiments, the PDE in question is the heat equation, such that the first step of MMS amounts to choosing some convenient function T_{ref} to act as the exact temperature profile. We then split the second step into two parts: First, we choose convenient functions to describe all physical parameters of the heat equation except the heat generation rate P. Secondly, we compute P such that the heat equation on differential form[2] is satisfied for the already chosen physical parameters and the chosen solution T_{ref}. In this way, the use of MMS allows us to obtain exact reference solutions of the heat equation without using any computationally expensive high-fidelity numerical solvers.

All manufactured solutions T_{ref} used to conduct the present study are listed in Table 4.1 (for 1D experiments) and Table 4.2 (for 2D experiments), along with the chosen conductivity k and the required heat generation rate P. For simplicity, the remaining physical variables were all set to unity, and they are therefore not listed in the aforementioned tables.

Notice that all the manufactured solutions in Tables 4.1 and 4.2 are parametrized by a parameter α. As such, each manufactured solution can be considered to represent a family of time series, with each time series corresponding to a unique α. In an application context, different α-values may correspond to e.g. different operating conditions or different initial states. In our experiments, we consider a total of 22 α-values, distributed across three sets, $\mathcal{A}_{\mathrm{train}}$, $\mathcal{A}_{\mathrm{val}}$ and $\mathcal{A}_{\mathrm{test}}$, as defined in Table 4.3. Time series corresponding to $\alpha \in \mathcal{A}_{\mathrm{train}}$ or $\alpha \in \mathcal{A}_{\mathrm{val}}$ are both used in the DNN training process described in Section 4.1.3. The remaining time series, corresponding to $\alpha \in \mathcal{A}_{\mathrm{test}}$, are used for evaluating the PBM, DDM and HAM models as described in Section 4.1.5. Observe that, of the four α-values in $\mathcal{A}_{\mathrm{test}}$, two lie within the interval $[0.1, 2.0]$ covered by $\mathcal{A}_{\mathrm{train}}$ while two do not. This choice of $\mathcal{A}_{\mathrm{test}}$ allows us to evaluate the generalizability of our models in both interpolation and extrapolation scenarios. We also highlight that, in each experiment, only individual manufactured solutions are considered. Thus, in any given experiment, 16 time series are used for DNN training, 2 time series are used for DNN validation, and 4 time series are used for model evaluation.

Since both the FVMs and DNNs operate on discrete data, it is necessary to discretize the continuous time series discussed above over some fixed spatial and temporal domains.

[1]Note however that $\boldsymbol{b}$ in Equations (4.3) and (4.5) will still be different than $\boldsymbol{b}$ in Equation (4.1) if $\boldsymbol{T}_{\mathrm{p}}^{n} \neq \boldsymbol{T}_{\mathrm{h}}^{n}$, despite the same definition being used to compute them.

[2]In principle, it is possible to use the integral form as well, but doing so makes the required calculations more complicated.

Table 4.1.: Manufactured solutions T_{ref} used for our experiments in 1D. Each solution is taken to be defined on the spatial domain $[0\,\text{m}, 1\,\text{m}]$ and the temporal domain $[0\,\text{s}, 5\,\text{s}]$. P and k are given in their respective SI units, while T is given in degrees Celsius. This table is duplicated in landscape mode in Appendix H for improved readability in print.

Label	$T_{\text{ref}}(x,t;\alpha)$	$P(x,t;\alpha)$	$k(x,t;\alpha)$
d1	$\alpha\left(t+\frac{1}{2}x^2\right)$	0	1
P1	$t+\frac{1}{2}\alpha x^2$	$1-\alpha$	1
P2	$\sqrt{t+\alpha+1}+10x^2(x-1)(x+2)$	$\frac{1}{2\sqrt{t+\alpha+1}}-120x^2-60x+40$	1
P3	$2+\alpha(x-1)\tanh\left(\frac{x}{t+0.1}\right)$	$\frac{\alpha}{(t+0.1)^2}\left(x(1-x)+2\left((x-1)\tanh\left(\frac{x}{t+0.1}\right)-t-0.1\right)\right)\operatorname{sech}^2\left(\frac{x}{t+0.1}\right)$	1
P4	$1+\sin(2\pi t+\alpha)\cos(2\pi x)$	$2\pi\left(\cos(2\pi t+\alpha)+2\pi\sin(2\pi t+\alpha)\right)\cos(2\pi x)$	1
k1	$t+\alpha x$	$1-\alpha$	$1+x$
k2	$5-\frac{\alpha x}{1+t}$	$\frac{\alpha(x-\alpha)}{(1+t)^2}$	$5-\frac{\alpha x}{1+t}$
k3	$e^{-t}\cdot\begin{cases}\alpha+2x, & x\le 0.5\\ \alpha+0.75+0.5x, & x>0.5\end{cases}$	$-e^{-t}\cdot\begin{cases}\alpha+2x, & x\le 0.5\\ \alpha+0.75+0.5x, & x>0.5\end{cases}$	$\begin{cases}0.5, & x\le 0.5\\ 2, & x>0.5\end{cases}$
k4	$4x^3-4x^2+\alpha(t+1)$	$\alpha-36x^2-8x+8$	$1+x$
A	$\sqrt{t+\alpha+1}+7x^2(x-1)(x+2)$	$\frac{1}{2\sqrt{t+\alpha+1}}-84x^2-42x+28$	1
B	$-\frac{x^3(x-\alpha)}{t+0.1}$	$\frac{x^4-\alpha x^3}{(t+0.1)^2}+\frac{12x^2-6\alpha x}{t+0.1}$	1

Table 4.2.: Manufactured solutions T_{ref} used for our experiments in 2D. Each solution is taken to be defined on the spatial domain $[0\,\text{m}, 1\,\text{m}] \times [0\,\text{m}, 1\,\text{m}]$ and the temporal domain $[0\,\text{s}, 5\,\text{s}]$. P and k are given in their respective SI units, while T is given in degrees Celsius. This table is duplicated in landscape mode in Appendix H for improved readability in print.

Label	$T_{\text{ref}}(x,y,t;\alpha)$	$P(x,y,t;\alpha)$	$k(x,y,t;\alpha)$
2P1	$t+0.5\alpha(x^2+y^2)+x$	$(1-2\alpha)$	1
2P2	$1+\sin(2\pi t+\alpha)\cos(2\pi x)\cos(2\pi y)$	$2\pi\cos(2\pi x)\cos(2\pi y)\left(\cos(2\pi t+\alpha)+4\pi\sin(2\pi t+\alpha)\right)$	1
2k1	$t+\alpha x+y^2$	$-(1+\alpha+2x+4y)$	$1+x+y$
2k2	$\alpha+(t+1)\cos(2\pi x)\cos(4\pi y)$	$\cos(2\pi x)\cos(4\pi y)\left(1+40\pi^2(t+1)\left(1+\sin(1\pi x)\sin(4\pi y)\right)\right)$	$2+\sin(2\pi x)\sin(4\pi y)$

In all experiments, we consider the temporal domain $[t_0, t_{\text{end}}] = [0\,\text{s}, 5\,\text{s}]$, which we split into $N_t = 5001$ equally spaced time levels, including the initial time level $n = 0$. The distance between two consecutive time levels is then $\Delta t = 0.001\,\text{s}$. The discretization of the spatial domain depends on the dimensionality of the problem at hand. For all experiments concerning 1D problems, we consider the spatial domain $[x_a, x_b] = [0\,\text{m}, 1\,\text{m}]$, which we split into N_j equally sized grid cells. In 2D, we consider instead the spatial domain $[x_a, x_b] \times [y_c, y_d] = [0\,\text{m}, 1\,\text{m}] \times [0\,\text{m}, 1\,\text{m}]$, which we split into N_j^2 equally sized grid cells. The value of N_j varies across experiments, and is therefore given in the introduction of each experiment. With the appropriate spatial discretization, we define grid nodes as in Section 2.2.3, and sample each manufactured solution T_{ref} at all grid nodes and time levels to define one *discrete* time series $\{\boldsymbol{T}_{\text{ref}}^n\}_{n=0}^{N_t-1}$ per value of α.

Table 4.3.: Parametrization: Selection of α-values corresponding to the training, validation and testing time series used in our experiments.

Purpose	Set of α-values	Symbol
Training	$\{0.1, 0.2, \ldots, 2.0\}\backslash\{0.7, 0.8, 1.1, 1.5\}$	$\mathcal{A}_{\text{train}}$
Validation	$\{0.8,\ 1.1\}$	$\mathcal{A}_{\text{val}}$
Testing	$\{-0.5, 0.7, 1.5, 2.5\}$	$\mathcal{A}_{\text{test}}$

Table 4.4.: The DNN hyperparameters used in our experiments.

Parameter	Value
Loss function	MSE
Learning rate	1e-5
Optimizer	Adam
Batch size	32
# hidden FC layers	4
Hidden FC layer width	80
LeakyReLU slope	0.01
Validation period	1e2
Overfit limit	20

4.1.3. DNN Training

Both DDM and CoSTA-based HAM require training of DNNs using the reference data described above. In accordance with the general discussion on DNN training in Section 2.3.3, one batch of data examples $(\boldsymbol{T}^n_{\text{ref}}, \boldsymbol{T}^{n+1}_{\text{ref}})$ (from the set of training data) is used to update the network's parameters per training iteration. The training procedure for both DNNs is illustrated in Figure 4.1a.[3] As illustrated, we use the mean squared error (MSE) cost function to measure the accuracy of the DNN predictions during training. For a DNN output vector $\boldsymbol{O}$ and a corresponding target output vector $\boldsymbol{O}_{\text{ref}}$, each with N_O components,[4] the MSE cost function is defined as

$$\text{MSE} = \sqrt{\frac{1}{N_O}\sum_{l=1}^{N_O}(O_l - (O_{\text{ref}})_l)^2}. \tag{4.6}$$

Based on the computed cost function, the Adam optimizer (Kingma and Ba, 2014) is used to update the network parameters at the end of each training iteration.

After a set period of training iterations (which we refer to as the "validation period"), the total MSE cost for all data examples in the *validation* set is computed. The validation cost is computed analogously to the training cost, but using all data examples from the validation data rather than a selection of data examples from the training data. We utilize the early stopping regularization technique by stopping the DNN training if a new lowest validation cost has not been recorded for a certain number of consecutive validation periods (this number is denoted "overfit limit" in Table 4.4).

4.1.4. DNN Architecture and Hyperparameters

Deep neural networks are used in both our DDM model and our HAM model. To ensure a fair comparison between the two models, we use the same DNN architecture and hyper-

[3]For simplicity, the figure assumes a batch size of 1. The true batch size used in our experiments is listed in Table 4.4. We also mention that, in accordance with current best practices, we normalize the temperature profiles in the training set to have a mean of zero and a standard deviation of 1 before using them to train the DNNs. All DNN inputs/outputs must therefore be normalized/unnormalized accordingly, but this is not illustrated to avoid cluttering the figure. We emphasize that the normalization coefficients are computed using the training data only, such as to avoid data leakage.

[4]For the DNN used for DDM, $\boldsymbol{O} = \boldsymbol{T}_d$ and $\boldsymbol{O}_{\text{ref}} = \boldsymbol{T}_{\text{ref}}$. For the DNN used for HAM, $\boldsymbol{O} = \hat{\boldsymbol{\sigma}}_{\text{NN}}$ and $\boldsymbol{O}_{\text{ref}} = \hat{\boldsymbol{\sigma}}$. For 1D problems, $N_O = N_j$, and for 2D problems, $N_O = N_j \cdot N_i$.

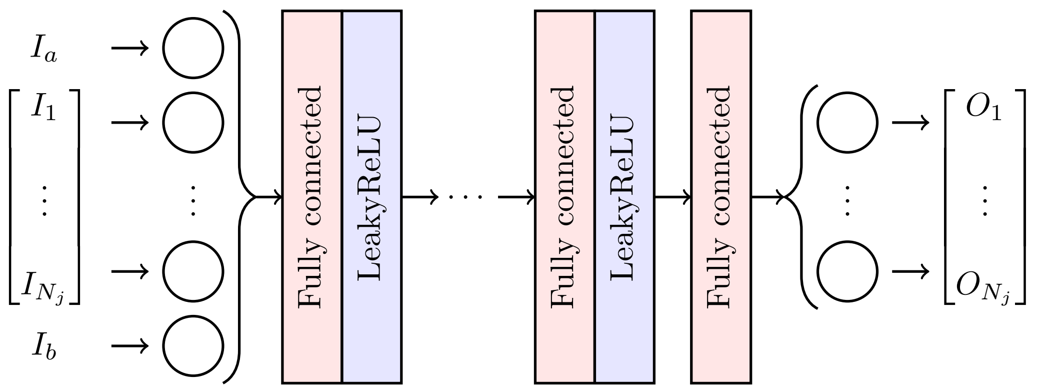


Figure 4.2.: The fully connected DNN architecture used in the 1D experiments of the present work. The same architecture is used for the 2D experiments, except the input and output layers are then larger to accommodate the increased number of components in the input and target output vectors. The definitions of the input vector $\boldsymbol{I}$ and the output vector $\boldsymbol{O}$ depend on whether the DNN is used for HAM or DDM. However, note that $\boldsymbol{I}$ always has more components than $\boldsymbol{O}$ since we only consider problems with known Dirichlet boundary conditions in this work.

parameters for both. For simplicity, we also keep the architecture[5] and hyperparameters the same across all experiments. Our chosen architecture, which is a fully connected architecture with LeakyReLU activation functions after each hidden layer, is illustrated in Figure 4.2. It was selected due to its simplicity, such that implementation details would not obscure the main focus of this thesis, which is the CoSTA framework. All relevant DNN hyperparameters are listed in Table 4.4. These parameter values were selected on the basis that we observed satisfactory and similar performance[6] for both DDM and HAM during a short series of preliminary experiments using these hyperparameters. These experiments were conducted in the same way as the experiments discussed in Section 4.2.2, but using different manufactured solutions (Solutions A and B from Table 4.1) to avoid data leakage. No further parameter tuning is required to answer our primary research questions, which are related to comparing functional and comparable – but not necessarily optimal – PBM, DDM and HAM models. Further research is required to define optimal DDM or HAM models for any given application, and some suggestions for such research are provided in Section 5.2.

4.1.5. Model Evaluation

In each numerical experiment, we consider one of the manufactured solutions listed in Table 4.1 or Table 4.2. In any given experiment, we aim to reproduce the time series $\{\boldsymbol{T}^n_{\text{ref}}\}_{n=0}^{N_t-1}$ for each $\alpha \in \mathcal{A}_{\text{test}}$ using our PBM, DDM and HAM models. We do this by first initializing all models to the true initial condition, i.e., we set $\boldsymbol{T}^0_{\text{p}} = \boldsymbol{T}^0_{\text{d}} = \boldsymbol{T}^0_{\text{h}} = \boldsymbol{T}^0_{\text{ref}}$. Then, we use the time stepping procedure illustrated in Figure 4.1b to generate the predicted time series $\{\boldsymbol{T}^n_{\text{p}}\}_{n=0}^{N_t-1}$, $\{\boldsymbol{T}^n_{\text{d}}\}_{n=0}^{N_t-1}$, and $\{\boldsymbol{T}^n_{\text{h}}\}_{n=0}^{N_t-1}$. To quantify the models' performance, we present the temporal development of the relative ℓ_2-errors,

[5]With the exception of the input and output layer size, which must be changed to accommodate the spatial discretization used in the individual experiments.

[6]By 'similar performance', we mean that the DNN validation losses recorded at the end of training was similar for DDM and HAM.

$\|\boldsymbol{T}_{\mathrm{p}}^n - \boldsymbol{T}_{\mathrm{ref}}^n\|_2/\|\boldsymbol{T}_{\mathrm{ref}}^n\|_2$, $\|\boldsymbol{T}_{\mathrm{d}}^n - \boldsymbol{T}_{\mathrm{ref}}^n\|_2/\|\boldsymbol{T}_{\mathrm{ref}}^n\|_2$, and $\|\boldsymbol{T}_{\mathrm{h}}^n - \boldsymbol{T}_{\mathrm{ref}}^n\|_2/\|\boldsymbol{T}_{\mathrm{ref}}^n\|_2$, of the PBM-, DDM- and HAM predictions with respect to the sampled manufactured solution $\boldsymbol{T}_{\mathrm{ref}}^n$. We also present the temperature profiles predicted by PBM, DDM and HAM at the final time level $n = N_t - 1$ (corresponding to $t = 5.0\,\mathrm{s}$) alongside the manufactured solution $T_{\mathrm{ref}}(x, t^{N_t-1}; \alpha)$. For completeness, we state that, for any vector $\boldsymbol{\omega} \in \mathbb{R}^{N_\omega}$, $N_\omega \in \mathbb{N}$, the ℓ_2-norm is defined as $\|\boldsymbol{\omega}\|_2 = \big(\sum_{l=1}^{N_\omega} \omega_l^2\big)^{1/2}$.

4.2. One-Dimensional Heat Transfer Experiments

In this section, we present and discuss the results of our numerical experiments on one-dimensional (1D) heat transfer problems. Three series of experiments with differing *a priori* physics knowledge are considered. From a PBM perspective, this means that we consider three different error sources. First, in Section 4.2.1, we consider experiments where all physics is known, such that any error in the PBM can be attributed to its numerical solver. Later, we consider scenarios where important physics is unknown, thereby resulting in modelling error. In Section 4.2.2, we consider experiments with an unknown source term P, while experiments with an unknown conductivity k are considered in Section 4.2.3.

4.2.1. Experiments without Modelling Error

In this section, we consider two experiments where all relevant physics is known. Comparing to the framework of Section 3.1, this means that we know the true operator $\mathcal{N}_\Omega$ and the true right-hand side function f. However, as we cannot solve the heat equation analytically in general, we must approximate $\mathcal{N}_\Omega$ with some numerical operator $\mathcal{N}_{\mathrm{num}}$, which corresponds to the Implicit Euler FVM. This introduces discretization error into the PBM and HAM models studied here.[7] In the first experiment, we study Solution d1 with $N_j = 20$, and in the second experiment, we study Solution P3 with $N_j = 200$. We expect PBM to perform well in these experiments, both in terms of accuracy and generalizability, considering that the FVM discretization is its only source of error. Furthermore, we expect PBM to be more accurate in the second experiment than in the first, since the more refined grid used in the second experiment will reduce the PBM's discretization error.[8]

We expect that it will be difficult for DDM to outperform PBM in the experiments considered here, as that would require the DDM to learn the full physics already instilled in the PBM *and* how to handle errors induced by the spatial and temporal discretizations. In comparison, the DNN in CoSTA-based HAM only has to learn how to correct the discretization error of the PBM. We therefore hypothesize that HAM will outperform DDM, and possibly also PBM, in the experiments of this section.

Below, we present and discuss the results of our 1D experiments without modelling error. We present the interpolation scenarios $\alpha \in \{0.7, 1.5\}$ first, and the extrapolation scenarios $\alpha \in \{-0.5, 2.5\}$ later.

[7]Recall that the DDM is completely independent of whatever *a priori* knowledge we assume to have (apart from the initial conditions and BCs), as it always learns the full dynamics from scratch. Hence it is never mentioned in discussions related to *a priori* knowledge.

[8]It should be noted that the PBM accuracy will also be the highest in the second experiment if Solution P3 is easier to model than Solution d1. However, looking at the analytic expressions for these solutions, as listed in Table 4.1, we see no reason to believe that this is the case. Therefore, we assume that any improved PBM accuracy in the second experiment can be attributed to the finer discretization.

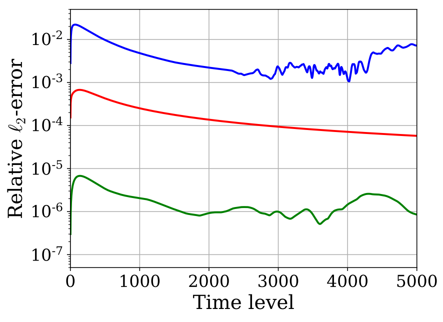
Interpolation Scenarios

The results for the experiment considering Solution d1 with $\alpha \in \{0.7, 1.5\}$ are presented in Figure 4.3. As expected, PBM gives predictions which are highly accurate and qualitatively indistinguishable from the true solutions (cf. Figures 4.3c and 4.3d). DDM also yields predictions which are good in accuracy, but they are still significantly less accurate than the PBM predictions (cf. Figures 4.3a and 4.3b), indicating that the DDM has not learnt the full physics of the problem. Lastly, CoSTA-based HAM, significantly outperforms both PBM and DDM, yielding errors that are several order of magnitude lower than the two other models. This indicates that the DNN-generated corrective source term of the HAM model is successful in correcting the discretization error of the Implicit Euler FVM. It also indicates that, in accordance with expectations, the learning task of the DNN in the HAM model is easier than that of the DNN in the DDM model.

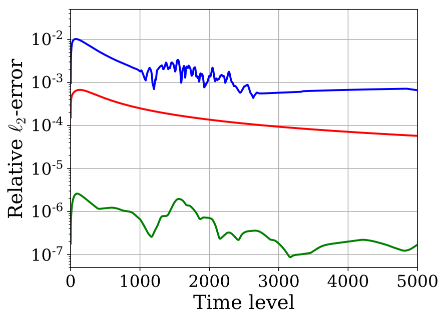
For the experiment considering Solution P3, whose results are presented in Figure 4.4, we observe that DDM performs moderately well with relative errors of a few percent. However, both PBM and HAM produce predictions which are several orders of magnitude more accurate than the DDM predictions (cf. Figures 4.4a and 4.4b). Note also that the PBM predictions are several orders of magnitude more accurate in this experiment than in the previous experiment, which is consistent with the finer spatial discretization used here. Even so, HAM still outperforms PBM, but the improvement is now much smaller than in the previous experiment. This could be due to Solution P3 being a more complex function of x and t than Solution d1. An important contribution to the discretization error of the PBM comes from the truncation errors of the finite difference approximations of the spatial and temporal derivatives in the heat equation. These truncation errors can generally be expressed using Taylor series expansions of the true solution T_{ref}. Thus, the complexity of T_{ref} affects the complexity of the truncation error, and thereby also the complexity of the discretization error of the PBM. Nonetheless, the DNN-generated corrective source term successfully reduces the discretization error in this experiment also.

It is worth noting that, in both experiments, we can observe a decreasing trend in the temporal development of the relative errors.[9] This trend, which is particularly prominent for PBM, may appear counter-intuitive as we naturally expect errors to accumulate across time levels. However, since we are considering relative errors, the accumulation of error in the numerical solutions is counteracted by the fact that $\|\boldsymbol{T}^n_{\text{ref}}\|_2$ is an increasing function of time for both solutions considered here. Furthermore, for Solution P3, another contributing factor is the fact that the true solution converges to a steady state. Thus, if a numerical solution converges to the correct steady state, but converges too slow or too fast, a significant error will be observed at the earliest time levels where the temperature changes are at their fastest. However, at later time levels, where both the true and numerical solutions are quite close to the steady state, their errors are expected to go to zero as they reach steady state. These factors explain the decreasing trends of the relative errors.

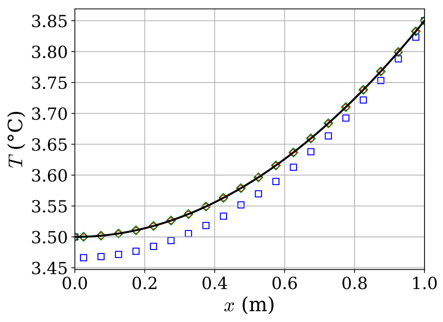
[9] With the exception of the DDM errors for Solution P3, which are roughly constant in time.



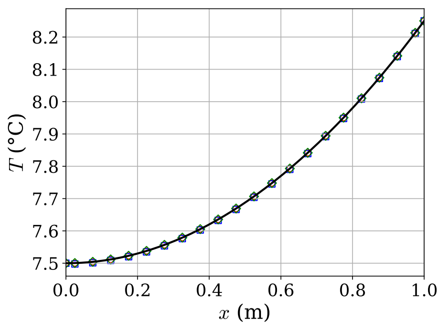
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.

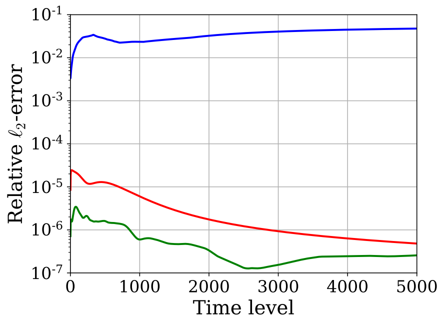


(c) $\alpha = 0.7$, time level $n = 5000$.

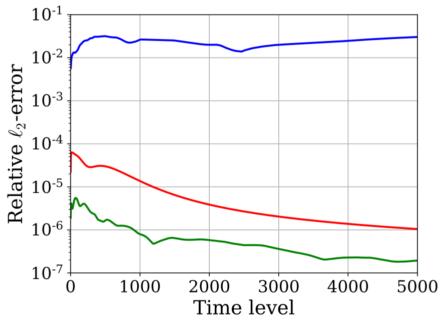


(d) $\alpha = 1.5$, time level $n = 5000$.

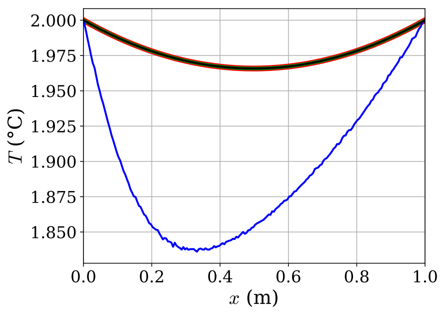
Figure 4.3.: Solution d1, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



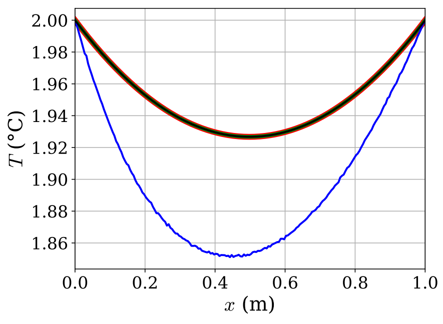
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.



(c) $\alpha = 0.7$, time level $n = 5000$.



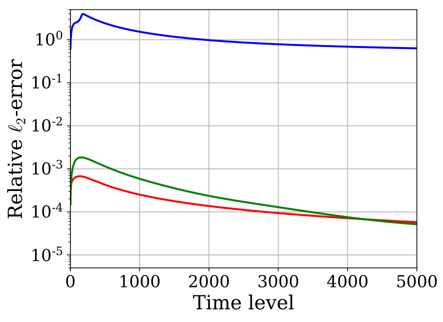
(d) $\alpha = 1.5$, time level $n = 5000$.

Figure 4.4.: Solution P3 with fine grid, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, — PBM, — DDM, — HAM).

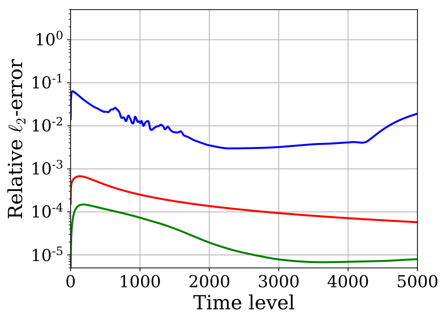
Extrapolation Scenarios

The results for the extrapolation scenarios $\alpha \in \{-0.5, 2.5\}$ are shown in Figure 4.5 for the experiment considering Solution d1, and in Figure 4.6 for the experiment considering Solution P3. Owing to its well-known, excellent generalizability, the PBM performs equally well in the extrapolation scenarios as in the interpolation scenarios considered previously. On the other hand, both DDM and HAM suffer a loss of accuracy in comparison to the interpolation scenarios. The loss of accuracy is particularly prominent for both solutions with $\alpha = -0.5$ (compare Figures 4.5a and 4.6a to e.g. Figures 4.3a and 4.4a), as both models see their DNN failing to generalize to this case. This is perhaps not surprising, since $\alpha = -0.5$ yields temperature profiles that are qualitatively different than those seen during training for both Solution d1 and Solution P3. Based on this observation, it is clear that both DDM and HAM would likely benefit from the use of data augmentation. As the applicability of data augmentation techniques vary from one problem to another (and even from one heat transfer problem to another), it is beyond the scope of the present work to consider data augmentation experimentally. However, we highly recommend exploring the use of data augmentation when using CoSTA-based HAM in any real-world application. Among possible techniques to consider, we mention inverting the spatial domain, as well as multiplicative and/or additive re-scaling of temperature profiles.

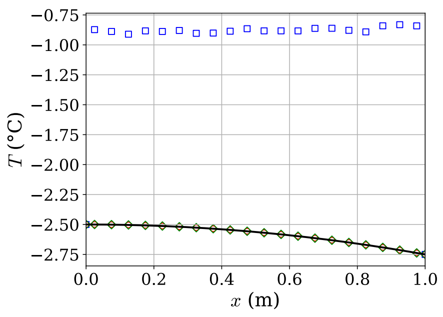
While HAM generally performs worse in the extrapolation scenarios than in the interpolation scenarios, it is still the most accurate model for both solutions with $\alpha = 2.5$ (cf. Figures 4.5b and 4.6b). Furthermore, HAM is roughly as accurate as PBM and roughly four orders of magnitude more accurate than DDM for Solution d1 with $\alpha = -0.5$ (cf. Figure 4.5a). In the final scenario, Solution P3 with $\alpha = -0.5$, PBM is the most accurate model (which means that the DNN-generated corrective source term of the HAM model is actually increasing the predictive error rather than reducing it). However, HAM is still several orders of magnitude more accurate than DDM (cf. Figure 4.6a). It is important to note that HAM's prediction at the final time level is also qualitatively correct, while the DDM prediction is not (cf. Figure 4.6c). Similarly, the HAM prediction for Solution d1 with $\alpha = -0.5$ is also qualitatively correct, while the DDM prediction is not (cf. Figure 4.5c). In conclusion, HAM is found to be more accurate and to generalize better than DDM. In comparing HAM to PBM, the situation is more nuanced, as HAM performs better than DDM for $\alpha = 2.5$, but worse for $\alpha = -0.5$. Overall, our results indicate that PBM generalizes slightly better than HAM, while both perform very well in terms of accuracy. When the interpolation cases are taken into account, HAM is generally more accurate than PBM. This indicates that the DNN of HAM has successfully learnt how to correct discretization errors in most scenarios considered.



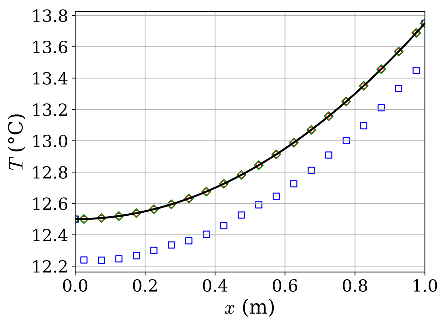
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.

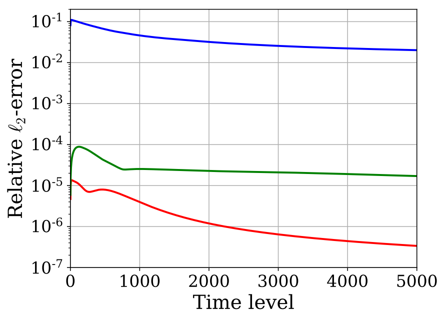


(c) $\alpha = -0.5$, time level $n = 5000$.

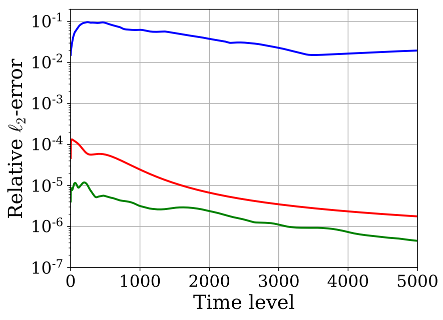


(d) $\alpha = 2.5$, time level $n = 5000$.

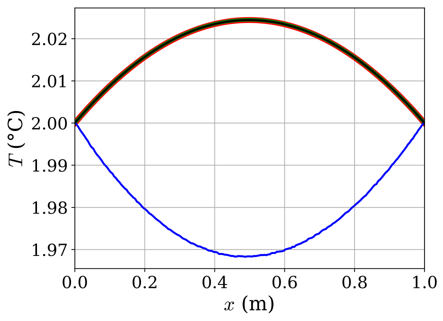
Figure 4.5.: Solution d1, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



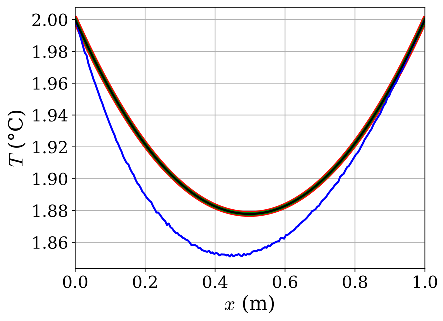
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.



(c) $\alpha = -0.5$, time level $n = 5000$.



(d) $\alpha = 2.5$, time level $n = 5000$.

Figure 4.6.: Solution P3 with fine grid, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, — PBM, — DDM, — HAM).

4.2.2. Experiments with Unknown Source Term

We now move on to the first series of experiments where the PBM and CoSTA-based HAM models are subject to modelling error. Here, we synthesize modelling error by assuming that the source term P in the heat equation is unknown. We therefore set $P = 0$ when calculating the vector $\boldsymbol{b}$ appearing in the FVM equations (4.1), (4.3) and (4.5). Comparing to the general framework of Section 3.1, this case corresponds to having an unknown f in Equation (3.1). Keep in mind that we still also have a discretization error (corresponding to using a numerical operator $\mathcal{N}_{\text{num}}$ to approximate $\mathcal{N}_{\Omega}$ in Equation (3.1)), such that the objective of the DNN-generated corrective source term in the HAM model is now to correct for both the unknown P and discretization errors.[10] Yet, this is still a more limited task than that of the DNN in DDM, which, as always, has to learn the full physics from data observations alone. We therefore hypothesize that DDM will be outperformed by HAM in these experiments. However, as the observational data implicitly inform the DDM of the unknown P, DDM may outperform PBM.

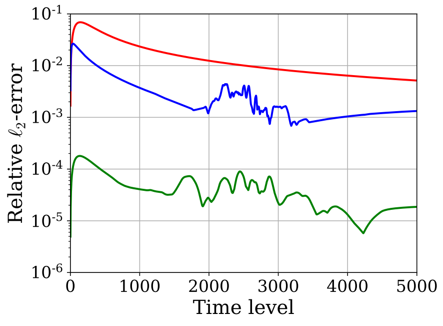
In this section, we consider four experiments, each based on one of the manufactured solutions P1–P4 (cf. Table 4.1). As stated above, we assume that the corresponding source terms P (listed in the third column of Table 4.1) are unknown. No further error is synthesized, and we utilize the exact boundary conditions in all models. The same spatial discretization with $N_j = 20$ equally sized grid cells was used in all four experiments. Below, we consider first the interpolation scenarios $\alpha \in \{0.7, 1.5\}$ for all four experiments, and then the extrapolation scenarios $\alpha \in \{-0.5, 2.5\}$ thereafter.

Interpolation Scenarios

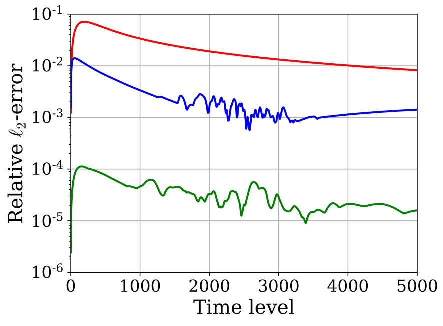
The interpolation scenario results for Solutions P1–P4 are presented in Figures 4.7–4.10. From these figures, we observe that PBM generally performs poorly because it is unable to model the unknown source term. While yielding decent predictions at the final time level for Solution P1 (cf. the bottom row of Figure 4.7), the final-time-level PBM predictions for all other solutions are qualitatively incorrect and/or yield a large relative error exceeding 10% (cf. Figures 4.8–4.10). DDM performs significantly better than PBM, and provides qualitatively correct predictions for Solutions P1, P2 and P4 (cf. the bottom rows of Figures 4.7, 4.8 and 4.10). This indicates that the DDM has successfully learnt both known and unknown physics in these experiments. However, in the experiment on Solution P3 (Figure 4.9), which appears to be the most difficult experiment considered in this section, only HAM is able to provide qualitatively correct predictions at the final time level. Furthermore, the HAM predictions are several orders of magnitude more accurate than the PBM and DDM predictions for the other three solutions.

Since DDM and HAM both use exactly the same DNN architecture with the same hyperparameters, the difference in performance between the two can only be attributed to the PBM incorporated in the HAM model. As discussed previously, the purpose of this PBM is to ease the learning task of the DNN by accounting for known physics such that the DNN does not have to learn *all* the relevant physics, as is the case for DDM. From the results presented here, it is clear that CoSTA-based HAM benefits from the PBM even when the PBM yields poor predictive accuracy when used on its own.

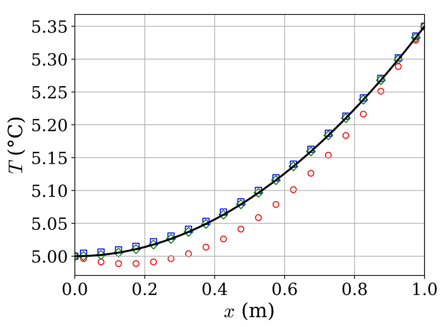
[10] In Appendix F, we discuss the relative impact of the discretization error and the modelling error on the PBM, and find that the modelling error is the dominant error for the four solutions considered in this section. Thus, if HAM significantly outperforms PBM here, we know that the DNN-generated corrective source term successfully corrects modelling error.



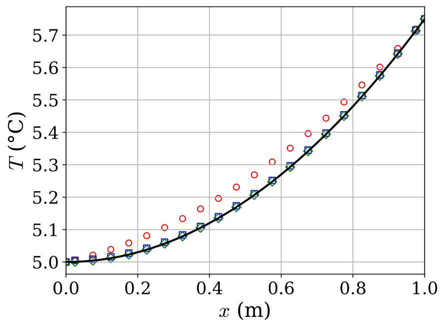
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.

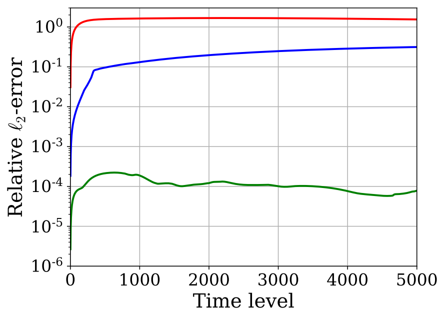


(c) $\alpha = 0.7$, time level $n = 5000$.

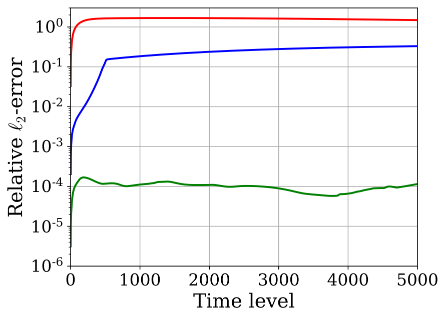


(d) $\alpha = 1.5$, time level $n = 5000$.

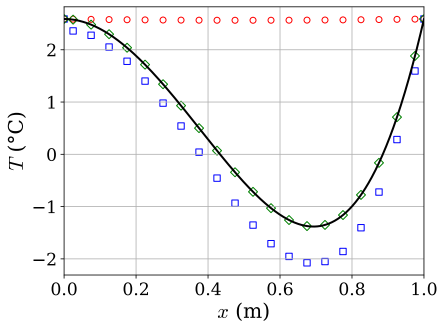
Figure 4.7.: Solution P1, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



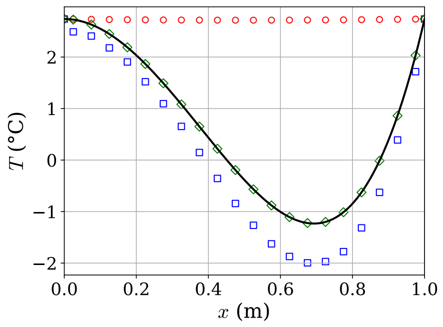
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.

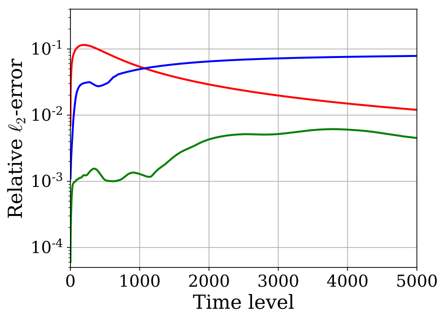


(c) $\alpha = 0.7$, time level $n = 5000$.

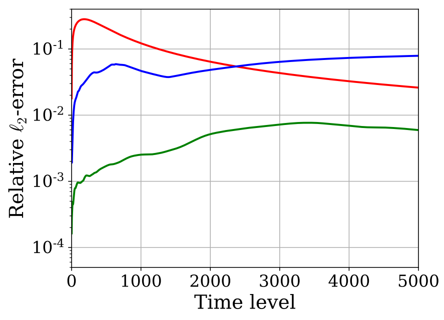


(d) $\alpha = 1.5$, time level $n = 5000$.

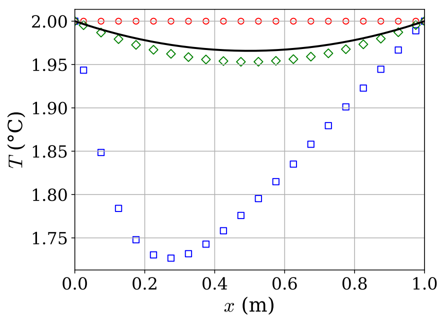
Figure 4.8.: Solution P2, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



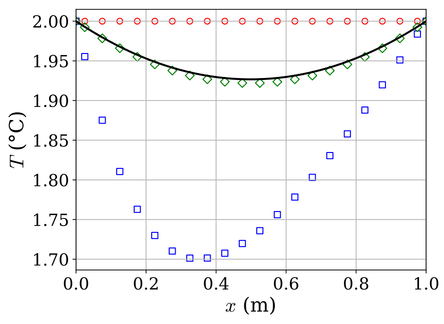
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.

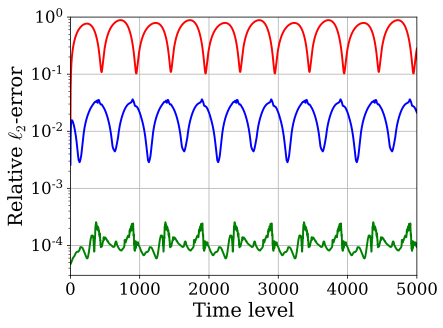


(c) $\alpha = 0.7$, time level $n = 5000$.

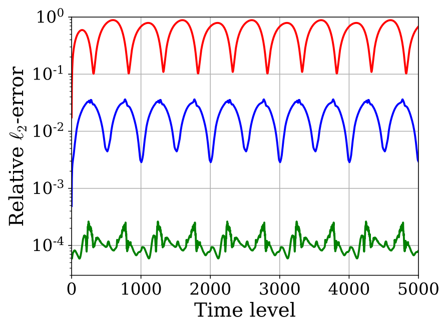


(d) $\alpha = 1.5$, time level $n = 5000$.

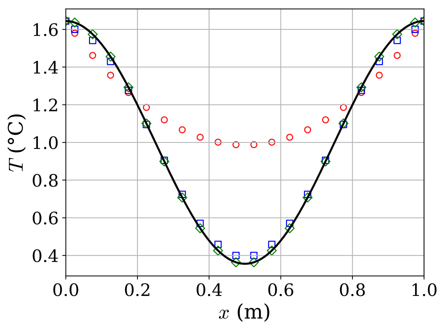
Figure 4.9.: Solution P3, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



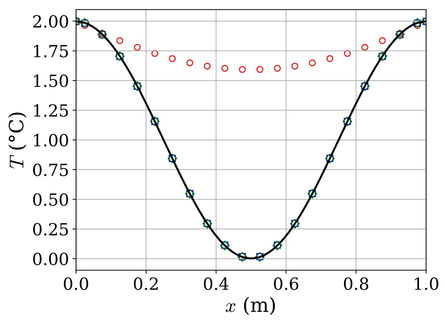
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.



(c) $\alpha = 0.7$, time level $n = 5000$.

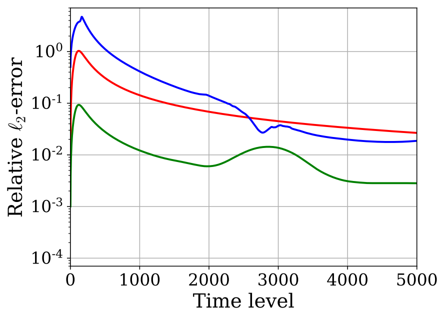


(d) $\alpha = 1.5$, time level $n = 5000$.

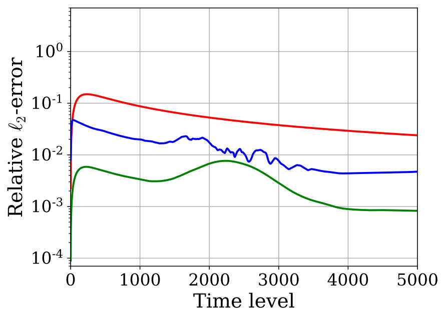
Figure 4.10.: Solution P4, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).

Extrapolation Scenarios

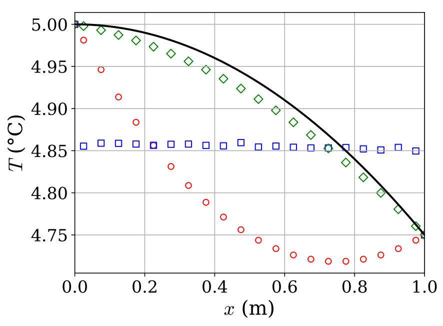
The results for Solutions P1–P4 with $\alpha \in \{-0.5, 2.5\}$ are shown in Figures 4.11–4.14. Just like in the experiments of Section 4.2.1, we observe that PBM performs similarly in the extrapolation scenarios as in the interpolation scenarios. However, as PBM generally exhibited poor accuracy in the interpolation scenarios, its accuracy leaves much to be desired also in the extrapolation scenarios. DDM performs significantly better than PBM, giving qualitatively correct predictions at the final time level for Solutions P2 and P4 (cf. the bottom rows of Figures 4.12 and 4.14). However, in terms of accuracy, HAM outperforms DDM for both these solutions (cf. the top rows of Figures 4.12 and 4.14). HAM also outperforms DDM for Solution P1, where the DDM prediction for $\alpha = -0.5$ is qualitatively incorrect while the HAM prediction is not (cf. Figure 4.11c). Furthermore, HAM is more accurate than both PBM and DDM for Solution P3 with $\alpha = 2.5$ (cf. Figure 4.13b). The remaining scenario – Solution P3 with $\alpha = -0.5$ – proves to be the most difficult scenario considered, as all three models fail to provide qualitatively correct predictions for this scenario (cf. Figure 4.13c). Overall, HAM is found to be the most accurate model in the scenarios considered here, followed by DDM and PBM.



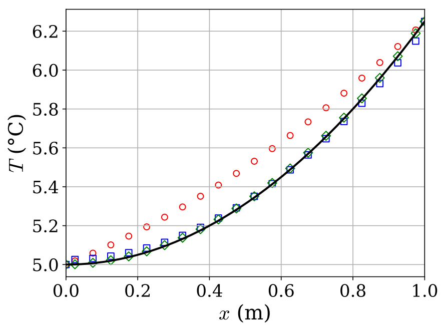
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.

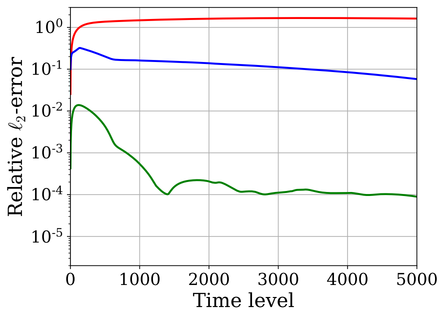


(c) $\alpha = -0.5$, time level $n = 5000$.

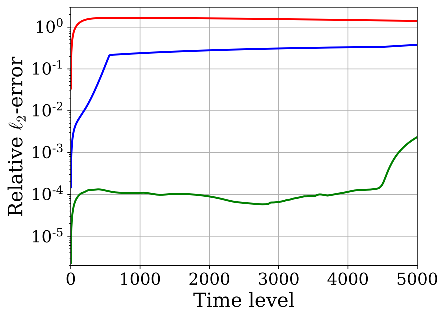


(d) $\alpha = 2.5$, time level $n = 5000$.

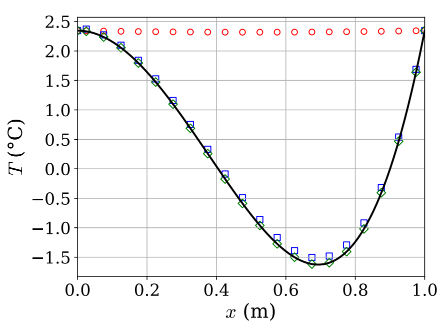
Figure 4.11.: Solution P1, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



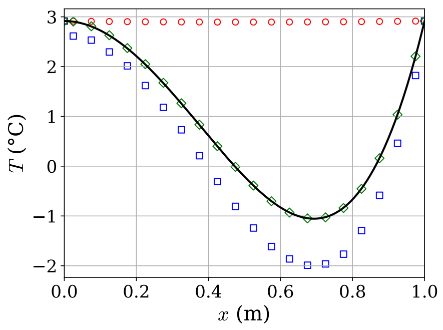
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.

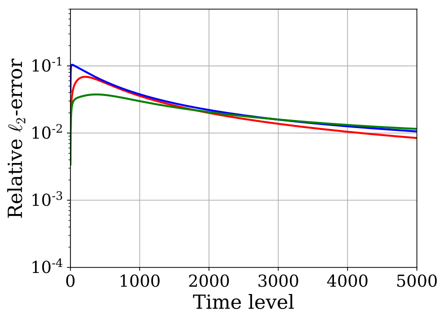


(c) $\alpha = -0.5$, time level $n = 5000$.

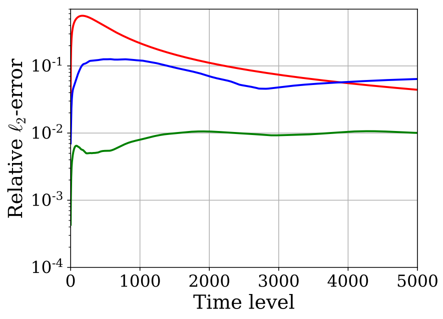


(d) $\alpha = 2.5$, time level $n = 5000$.

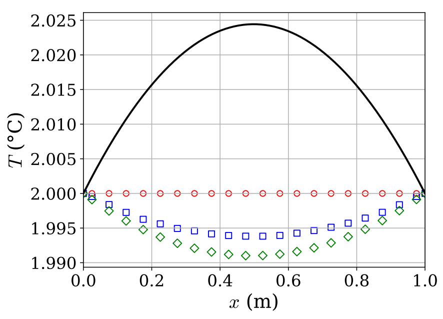
Figure 4.12.: Solution P2, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



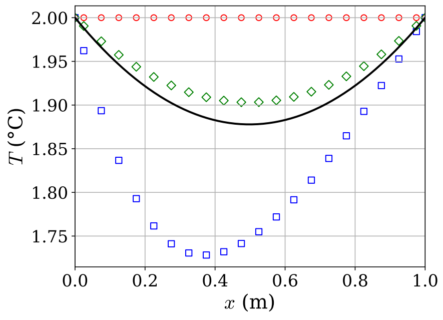
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.

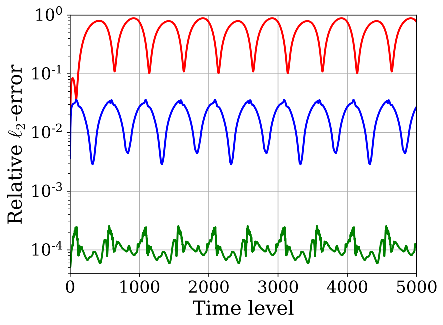


(c) $\alpha = -0.5$, time level $n = 5000$.

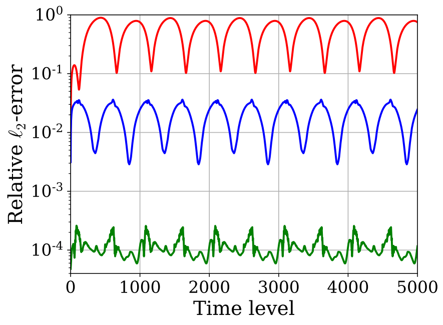


(d) $\alpha = 2.5$, time level $n = 5000$.

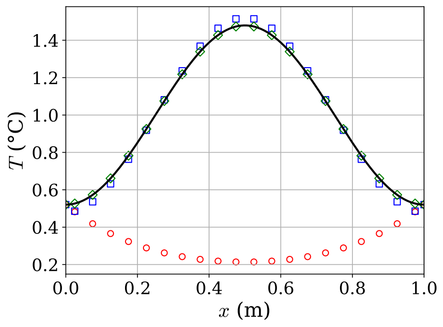
Figure 4.13.: Solution P3, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



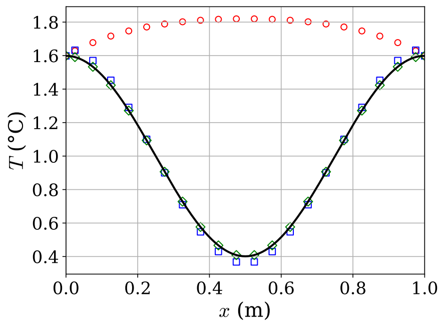
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.



(c) $\alpha = -0.5$, time level $n = 5000$.



(d) $\alpha = 2.5$, time level $n = 5000$.

Figure 4.14.: Solution P4, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).

4.2.3. Experiments with Unknown Conductivity

In this section, we will consider a second series of 1D experiments where the PBM and HAM models are affected by modelling error. This time, the modelling error stems from an unknown conductivity k, which is approximated by a constant unit conductivity. Comparing to the framework of Section 3.1, the case considered here corresponds to not knowing the true operator $\mathcal{N}_\Omega$ in the governing equation (3.1). This operator is therefore approximated by another operator $\widetilde{\mathcal{N}}$ in the governing equation of our PBM (3.3). As we still cannot solve the PBM's governing equation analytically, we must further approximate $\widetilde{\mathcal{N}}$ with a numerical operator $\widetilde{\mathcal{N}}_{\text{num}}$ to obtain our PBM predictions. Relating this back to the present experiments, $\widetilde{\mathcal{N}}_{\text{num}}$ corresponds to the Implicit Euler FVM, which is utilized in Equations (4.1), (4.3) and (4.5). We use $k = 1$ when calculating $\mathbb{A}$ and $\boldsymbol{b}$ in all three of these equations.

The manufactured solutions considered in the experiments of this section are Solutions k1–k4, and we use a spatial discretization with $N_j = 50$ grid cells. These are all listed in Table 4.2, along with their corresponding conductivities k (which are assumed unknown here) and their corresponding heat generation rates P (which are considered *known* here, in contrast to the previous experiment series). From an application perspective, Solutions k2 and k3 are particularly interesting due to their physical interpretation. For Solution k2, the conductivity is proportional to the temperature, which holds true for so-called Fermi gases. As discussed by Kittel (2005, chapter 6), the Fermi gas model is useful for modelling the physical properties of metals. For Solution k3, we have a piecewise constant conductivity, which can be thought of as a rudimentary model for two adjacent materials with different conductivities. Such a scenario is relevant e.g. for modelling heat transfer through walls, which are seldom made from just one material.

As before, the results for the interpolation scenarios $\alpha \in \{0.7, 1.5\}$ are presented first, while the extrapolation scenarios $\alpha \in \{-0.5, 2.5\}$ are considered later. Just like in the previous series of experiments, we expect DDM and HAM to outperform PBM because both these models can learn about the unknown conductivity from observations, while PBM cannot. We also hypothesize that CoSTA-based HAM will continue to outperform DDM, due to its use of PBM to presumably make the DNN's learning task easier.

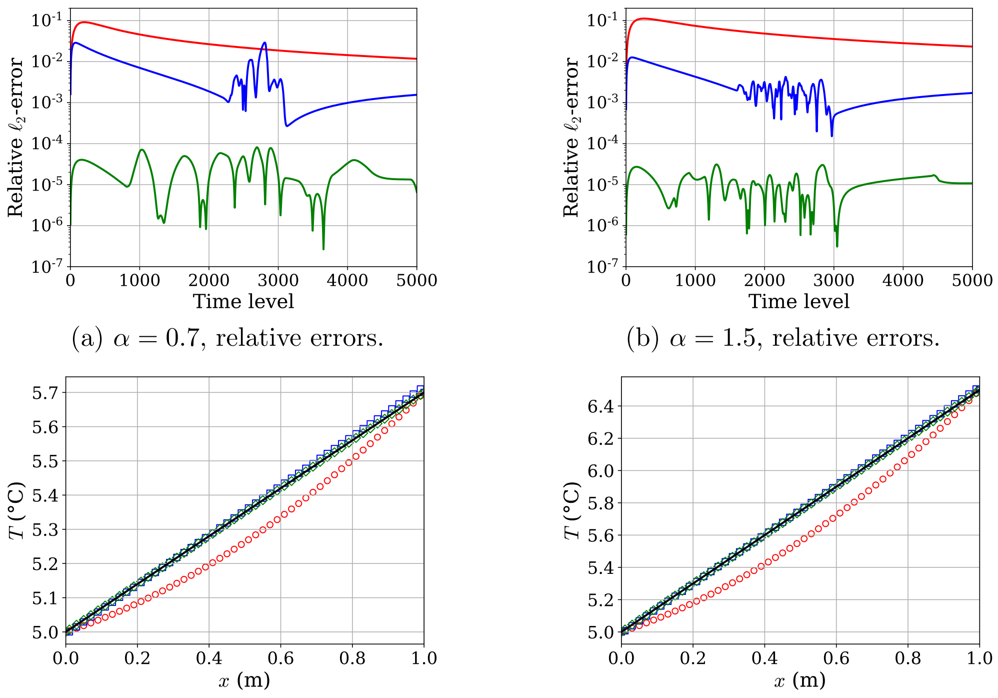
Interpolation Scenarios

Figures 4.15–4.18 display the interpolation scenario results for Solutions k1–k4, respectively. For Solution k1 (cf. Figure 4.15), we observe that the PBM predictions are correct near the boundaries since the boundary conditions are known. However, as the temperature increases as a function of time, the PBM predictions lag behind the true solutions in the middle of the domain, causing significant errors in that region. Looking at Figures 4.15c and 4.15d, it is clear that both DDM and HAM are significantly more accurate than PBM, as both (approximately) capture the linear nature of Solution k1. Furthermore, from Figures 4.15a and 4.15b, we observe that the HAM predictions are several orders of magnitude more accurate than those of the DDM.

Interestingly, it appears that all three models are able to accurately model Solution k2 (cf. Figure 4.16). This is not so surprising for the DDM; it is only trained to learn the true temperature profile, which is linear in both time and space and therefore presumably quite easy to learn. It is perhaps more surprising that the PBM performs as well as it does, but knowledge of the correct conductivity is evidently not necessary for the PBM to provide qualitatively correct predictions for this solution, as can be seen from the bottom row of Figure 4.16. Since the PBM performs rather well, it is not surprising that the HAM model also has high predictive accuracy. Still, it is remarkable that HAM yet again provides predictions whose relative ℓ_2-error is consistently at least one order of magnitude lower than for the other models (cf. the top rows of Figure 4.16).

There are much greater performance differences between the models for Solution k3 (cf. Figure 4.17. Unsurprisingly, PBM does not capture the discontinuity in the conductivity at the domain center, and thereby predicts a smooth temperature profile. In contrast, both HAM and DDM learn about the discontinuity from data observations, as can be seen from the bottom row of Figure 4.17. However, only HAM is able to give an accurate prediction of the temperature profile at the final time level. Looking at the development of the ℓ_2-errors in the top row of Figure 4.17, it appears that DDM starts off making quite accurate predictions, but that a small error in the solution has amplified and accumulated over time. As the DNN of the data-driven model is trained using error-free reference data only, it is not trained to correct for errors in its input. Therefore, when presented with input that has errors in it, this error might as well be amplified in the DNN's output. When this output is then used as input at the next time level, it is easy to see how errors can accumulate rather quickly. As our DDM does not have any mechanism to avoid this kind of error inflation, its predictions could in principle diverge all the way to numerical overflow. Such divergent behaviour could in principle also occur for the DNN in CoSTA-based HAM models, although it is not observed in the experiments of the present work. Still, this is a theoretical weak-point of CoSTA-based HAM (or of our realization of the approach, at least) which should be addressed in future research, as we highlight in Section 5.2.

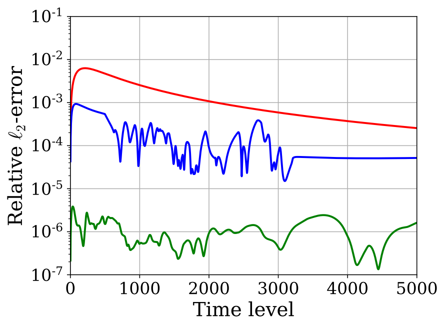
The results for Solution k4 tell much the same story as those for Solution k1. From the bottom row of Figure 4.18, we see that both DDM and HAM provide qualitatively good predictions, while PBM has significant error in the middle of the domain. Furthermore, HAM is significantly more accurate than DDM. An interesting feature of the ℓ_2-error curves in the bottom row of Figure 4.18 is the presence of a large spike in each of the DDM error curves. Similar spikes can be seen in other error curves as well, but the ones shown here are the most prominent so far. Such spikes occur when a model's predictions transition from being too hot to being too cold, or vice versa. Similarly, smaller and more frequent spikes correspond to oscillations about the true solution in parts of the spatial domain, while the predictive error is more stable in other regions. As can be seen from all ℓ_2-error curves presented so far in this chapter, both DDM and HAM are susceptible to significant fluctuations in the temporal development of the ℓ_2-error of their predictions. This can be explained by the fact that their DNNs are not trained to output temporally coherent data. During training, they are simply given randomly selected temperature profiles and tasked with predicting the corresponding subsequent temperature profiles. This issue could be addressed e.g. by using recurrent neural networks or temporal convolutional networks, which are specifically designed to handle and take advantage of temporal correlations in time series (cf. Section 2.3.1).



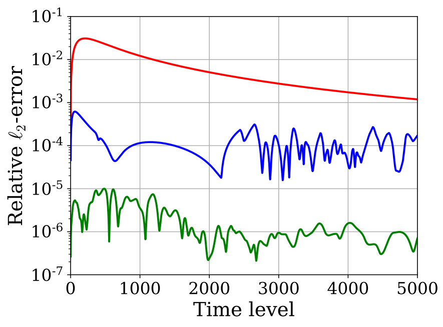
(a) $\alpha = 0.7$, relative errors. (b) $\alpha = 1.5$, relative errors.

(c) $\alpha = 0.7$, time level $n = 5000$. (d) $\alpha = 1.5$, time level $n = 5000$.

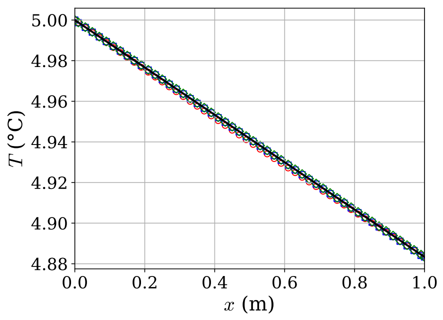
Figure 4.15.: Solution k1, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



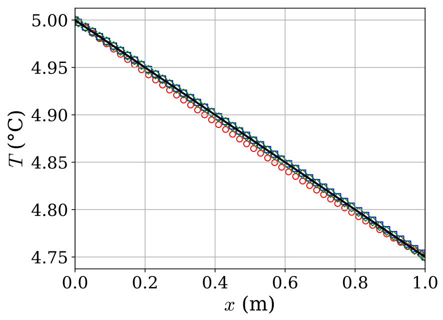
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.

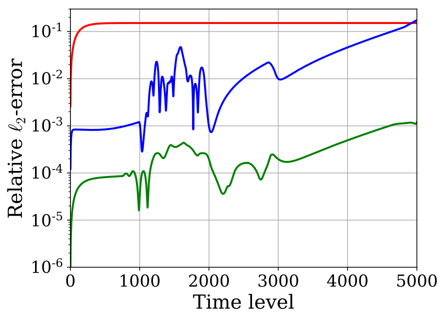


(c) $\alpha = 0.7$, time level $n = 5000$.

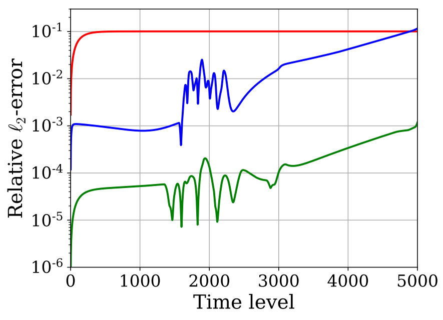


(d) $\alpha = 1.5$, time level $n = 5000$.

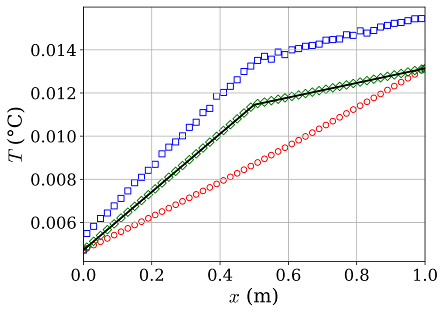
Figure 4.16.: Solution k2, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



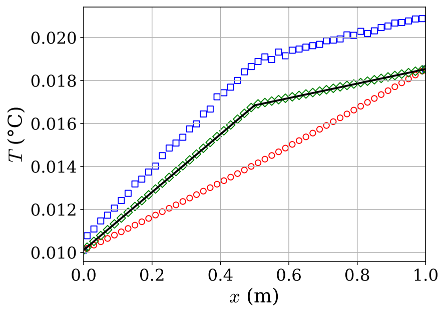
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.

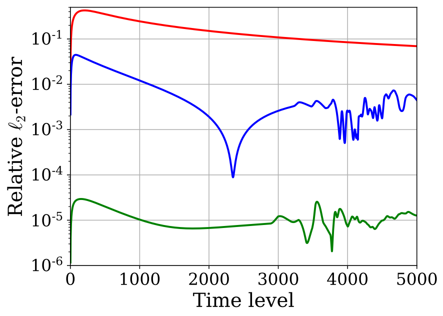


(c) $\alpha = 0.7$, time level $n = 5000$.

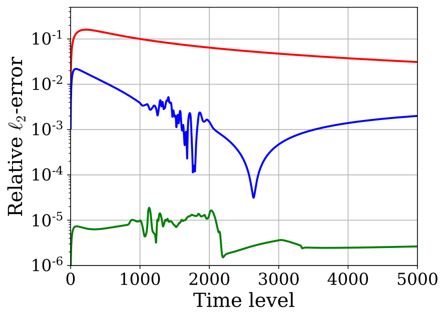


(d) $\alpha = 1.5$, time level $n = 5000$.

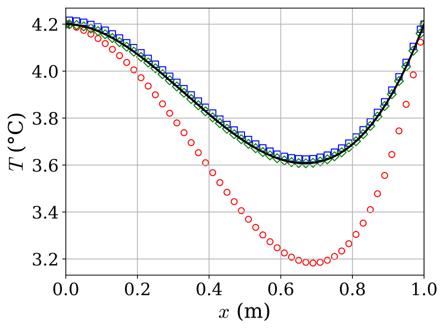
Figure 4.17.: Solution k3, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



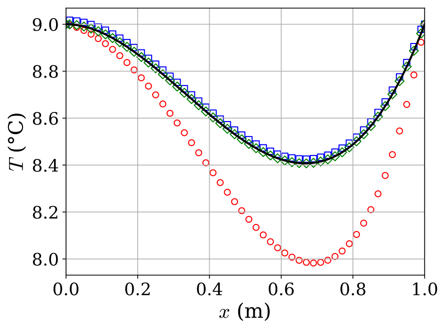
(a) $\alpha = 0.7$, relative errors.



(b) $\alpha = 1.5$, relative errors.



(c) $\alpha = 0.7$, time level $n = 5000$.



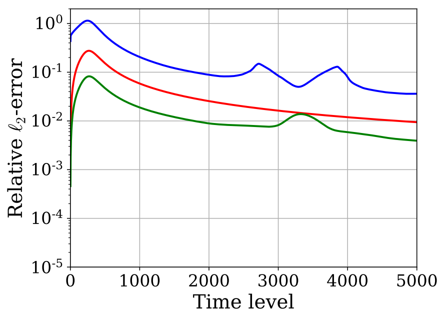
(d) $\alpha = 1.5$, time level $n = 5000$.

Figure 4.18.: Solution k4, interpolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{0.7, 1.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).

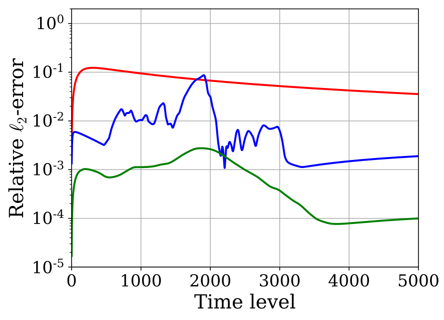
Extrapolation Scenarios

We present the extrapolation scenario results for Solutions k1–k4 in Figures 4.19–4.22. As in the previous two experiment series, we observe that PBM generalizes very well; its predictive accuracy is virtually unchanged from the interpolation scenarios to the extrapolation scenarios for all four solutions considered here (compare Figures 4.19–4.22 to Figures 4.15–4.18). In contrast, DDM struggles significantly in the extrapolation scenarios, providing qualitatively incorrect predictions for both Solution k1 and Solution k2 with $\alpha = -0.5$ (cf. Figures 4.19c and 4.20c). Additionally, for Solution k3 and Solution k4 with $\alpha = -0.5$, the relative ℓ_2-error of the DDM predictions at the final time level far exceeds 10%, which is poor (cf. Figures 4.21a and 4.22a). Furthermore, it is worth noting the that DDM continues to struggle with the same error amplification issues for Solution k3 as in the interpolation scenarios. A similar phenomenon also appears to be occurring for Solution k4 in the extrapolation scenarios considered here (cf. Figure 4.22).

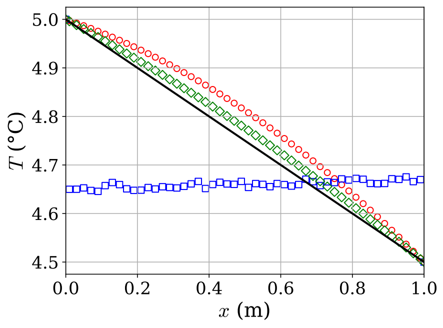
The HAM model does not struggle with generalization in the same way as the DDM, as it provides qualitatively correct predictions for all four solutions with both $\alpha = -0.5$ and $\alpha = 2.5$. The HAM and PBM models are roughly equally accurate for solutions k1 and k2 with $\alpha = -0.5$, while HAM is clearly the most accurate model for all other scenarios. Thus, HAM once again demonstrates superior accuracy paired with great generalizability.



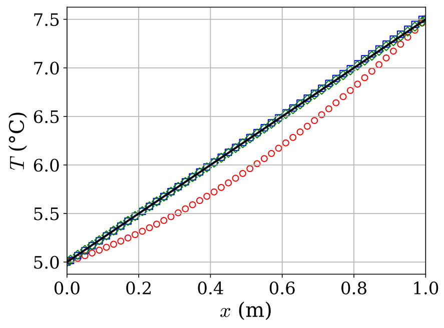
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.

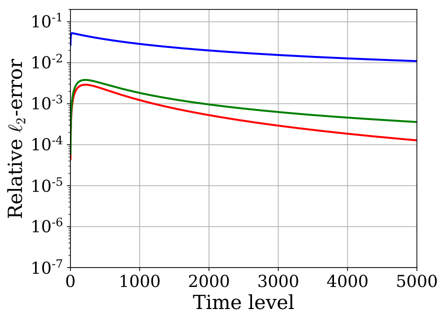


(c) $\alpha = -0.5$, time level $n = 5000$.

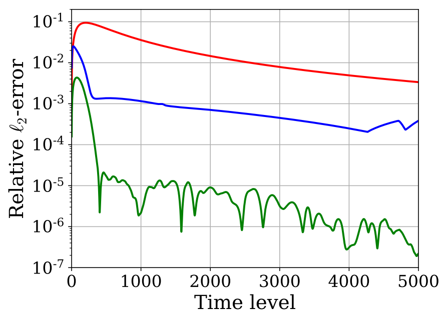


(d) $\alpha = 2.5$, time level $n = 5000$.

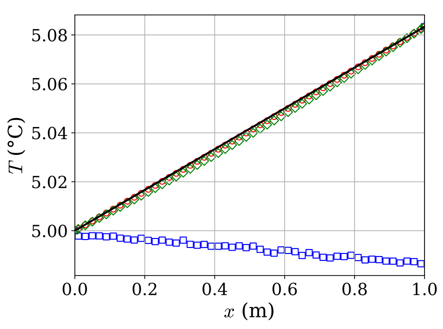
Figure 4.19.: Solution k1, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



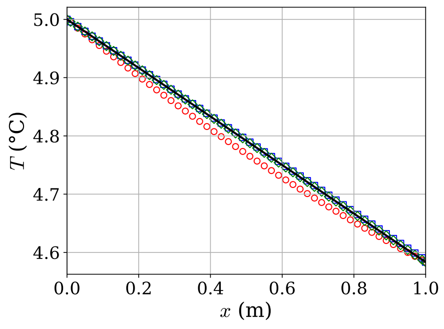
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.

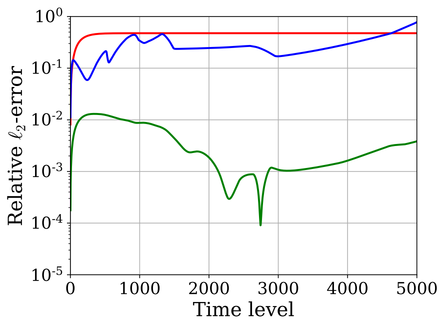


(c) $\alpha = -0.5$, time level $n = 5000$.

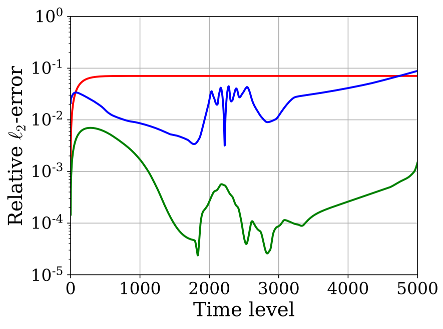


(d) $\alpha = 2.5$, time level $n = 5000$.

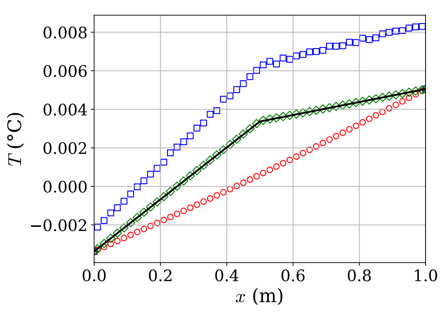
Figure 4.20.: Solution k2, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



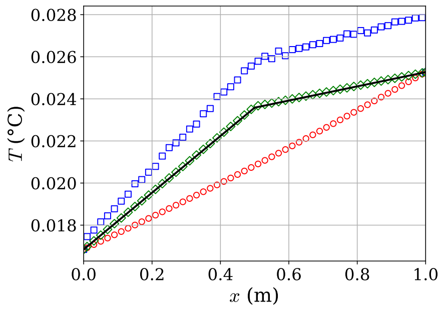
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.

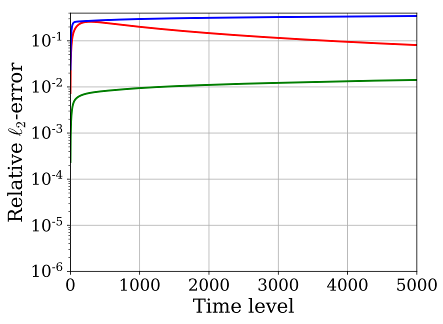


(c) $\alpha = -0.5$, time level $n = 5000$.

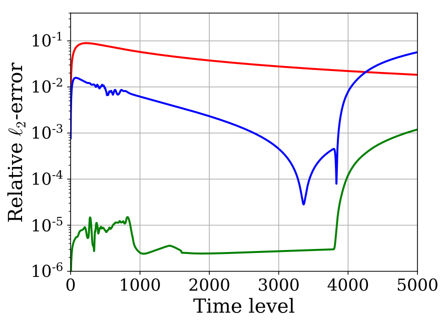


(d) $\alpha = 2.5$, time level $n = 5000$.

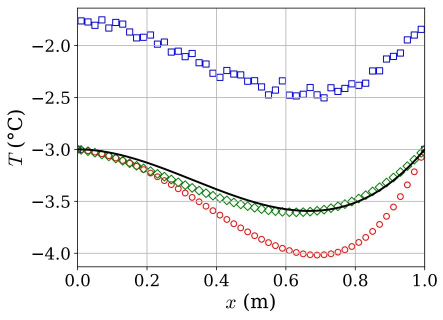
Figure 4.21.: Solution k3, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).



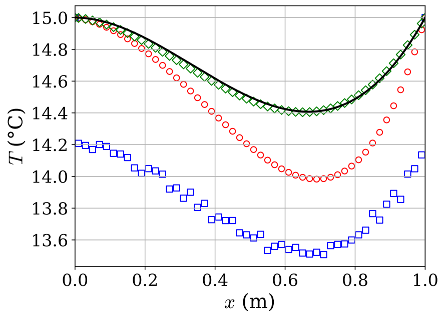
(a) $\alpha = -0.5$, relative errors.



(b) $\alpha = 2.5$, relative errors.



(c) $\alpha = -0.5$, time level $n = 5000$.



(d) $\alpha = 2.5$, time level $n = 5000$.

Figure 4.22.: Solution k4, extrapolation: Comparison of relative ℓ_2-errors and final temperature profiles for $\alpha \in \{-0.5, 2.5\}$ (— Exact, ○ PBM, □ DDM, ◇ HAM).

4.2.4. Summary and Further Discussion of Experiments in One Dimension

In this section, we have considered three series of experiments where the predictive accuracy and generalizability of PBM, DDM and CoSTA-based HAM models is investigated. The experiments are all based on one-dimensional heat transfer problems where varying degrees of *a priori* knowledge is assumed. In the first series of experiments, described in Section 4.2.1, we assumed that the full physics were known, such that the discretization error was the only error present in the physics-based predictions. For the remaining two series, modelling error was synthesized as we either assumed zero internal heating/cooling ($P = 0$, cf. Section 4.2.2) or a constant unit conductivity ($k = 1$, cf. Section 4.2.3).

HAM significantly outperforms PBM and DDM in terms of accuracy in all experiments. This holds true even when both PBM and DDM perform poorly on their own, such as in the experiment considering Solution P1 with $\alpha = -0.5$ (cf. Figure 4.11c). Thus, the hybrid model is in some sense greater than the sum of its parts. In terms of generalizability, PBM is virtually unbeatable, as it generalizes perfectly to any scenario where its underlying assumptions hold. Due to the bias-variance trade-off[11] affecting its DNN, HAM does not generalize quite as well as PBM, but still offers impressive generalizability even in the extrapolation scenarios. In fact, HAM only fails to provide qualitatively correct predictions for one scenario in one experiment (cf. Figure 4.13c). It should also be mentioned that, due to modelling error, the PBM prediction is qualitatively incorrect for that scenario as well. Overall, the generalizability of the HAM model is far superior to that of the DDM. Thus, the results discussed here indicate that there is significant merit to the concept of combining PBM and DDM in a single CoSTA-based HAM model.

While the results of this section are indeed promising, it is important to keep in mind that we have identified some important weak-points of CoSTA-based HAM. First of all, we have observed a lack of coherence in the temporal development of the ℓ_2-errors of the HAM (and DDM) predictions. This comes as a direct result of our choice of DNN architecture and training routine. A simple fully-connected neural network trained in the way we described in Section 4.1 has no way of learning that there is a temporal correlation between certain data samples. This is indeed why other types of DNNs such as recurrent neural networks have been popular for time series forecasting over the last decades (cf. Section 2.3). As such, the experiments indicate that the DNN we have used, which was chosen to be as simple as possible for the purpose of not obscuring the CoSTA framework itself, has compromised the accuracy of our models to some extent. This is not to say that our comparison of HAM and DDM is unfair or invalid, as they both use the same sub-optimal DNN. However, it does indicates that one should probably implement a different DNN if using CoSTA-based HAM for any real-world application.

Another important observation, which also relates to the DNN, is the sign of divergent behaviour observed for the DDM. Although we have not observed equally strong signs for HAM, we currently have no reason to assume that HAM cannot exhibit such behaviour as well. The main issue is that we do not know *a priori* how errors in the prediction at one time level will affect predictions at later time levels. Ideally, we would want the DNN to ignore any features in its input that stem from errors made at earlier time levels. In practice, this is difficult, as it would require the DNN to be able to both recognize and correct its *own* errors. If it was generally possible for a DNN to do this, then simply

[11]The bias-variance trade-off essentially states that when optimizing a DNN for accuracy on a finite set of training data, the DNN becomes biased towards the training data. This bias negatively affects DNN accuracy on data not in the training set. Maier (2020) provides an excellent introduction to the bias-variance trade-off.

applying the part of the DNN responsible for the error recognition and correction to the DNN's output would yield error-free predictions for all problems in all scenarios always. Such a perfect DNN does not exist, due to the bias-variance trade-off. One possible option is to explore adversarial training (Goodfellow et al., 2014), which should reduce the influence of small errors on the DNN's predictions. While adversarial training would likely reduce the probability of divergent behaviour by making the DNN more robust, it does not provide any stability guarantees. This is because adversarial training does not provide any bound on error accumulation across multiple time levels. If we want to ensure that our HAM model cannot exhibit divergent behaviour, we therefore have to look elsewhere.

This problem of handling errors in past predictions is conceptually similar to the topic of stability analysis in numerical mathematics. When conducting a stability analysis, one is interested in determining if a numerical scheme amplifies or attenuates rounding errors. Mathematically, there is no difference between rounding errors and errors stemming from sub-optimal DNN predictions at past time levels. This suggests that traditional stability analysis techniques may provide some insight into the long-term behaviour of the CoSTA-based model. However, applying these techniques to hybrid models is not straight-forward, because the mathematical properties of the mapping parametrized by the DNN are generally unknown. Investigating how traditional stability analysis techniques can be extended or adapted for use with hybrid models is an interesting line of research which, if successful, would increase the trustworthiness of CoSTA-based HAM models.

Lastly, we highlight that in spite of the overall good performance of HAM in our experiments, it is still possible and desirable to improve HAM further, both in terms of accuracy and generalizability. The aforementioned change of DNN architecture may help in this respect. Another possibility worth exploring is data augmentation, as briefly discussed in Section 4.2.2.

4.3. Two-Dimensional Heat Transfer Experiments

We now move on to experiments concerning two-dimensional (2D) heat transfer problems. For brevity, we restrict ourselves to scenarios where modelling error is the dominant error source in the PBM. However, we still consider two different origins of the modelling error; in Section 4.3.1, the modelling error is caused by an unknown source term P, while in Section 4.3.2, it is caused by an unknown conductivity k. Neither HAM nor DDM have their DNN architecture of hyperparameters changed from the 1D experiments of the previous section, except that the dimensionality of the input and output layers must be increased to match the number of grid cells.

In the 1D experiments, we have presented the predicted solutions at the final time level in order to facilitate qualitative assessment of the PBM, DDM and HAM models. However, in 2D, the analogous contour plots of the predicted temperature fields were found to be largely uninformative, as only very large relative errors result in noticeable colour changes. In this section, we therefore present the relative *difference* between the predicted fields and the reference fields at the final time level. That is, we illustrate the relative differences $(\boldsymbol{T}_{\mathrm{p}}^{N_t-1} - \boldsymbol{T}_{\mathrm{ref}}^{N_t-1})/\boldsymbol{T}_{\mathrm{ref}}^{N_t-1}$, $(\boldsymbol{T}_{\mathrm{d}}^{N_t-1} - \boldsymbol{T}_{\mathrm{ref}}^{N_t-1})/\boldsymbol{T}_{\mathrm{ref}}^{N_t-1}$, and $(\boldsymbol{T}_{\mathrm{h}}^{N_t-1} - \boldsymbol{T}_{\mathrm{ref}}^{N_t-1})/\boldsymbol{T}_{\mathrm{ref}}^{N_t-1}$, where all subtractions and divisions are applied component-wise.[12] As

[12]For these illustrations, we use the imshow function of Matplotlib, which interpolates the discrete differences to produce smooth error fields.

before, we also present the temporal development of the relative ℓ_2-errors of the three models' predictions.

4.3.1. Experiments with Unknown Source Term

In this section, we consider two experiments where the source term of the heat equation is assumed unknown. As for the experiments in Section 4.2.2, we therefore use $P = 0$ when calculating $\boldsymbol{b}$ in Equations (4.1), (4.3) and (4.5). In the first experiment, we consider Solution 2P1, while Solution 2P2 is considered in the second experiment (cf. Table 4.2). For both experiments, we use $N_j = N_i = 20$, resulting in a total of $20^2 = 400$ grid cells. As in earlier experiments, we present the interpolation scenarios $\alpha \in \{0.7, 1.5\}$ first and the extrapolation scenarios $\alpha \in \{-0.5, 2.5\}$ thereafter.

Interpolation Scenarios

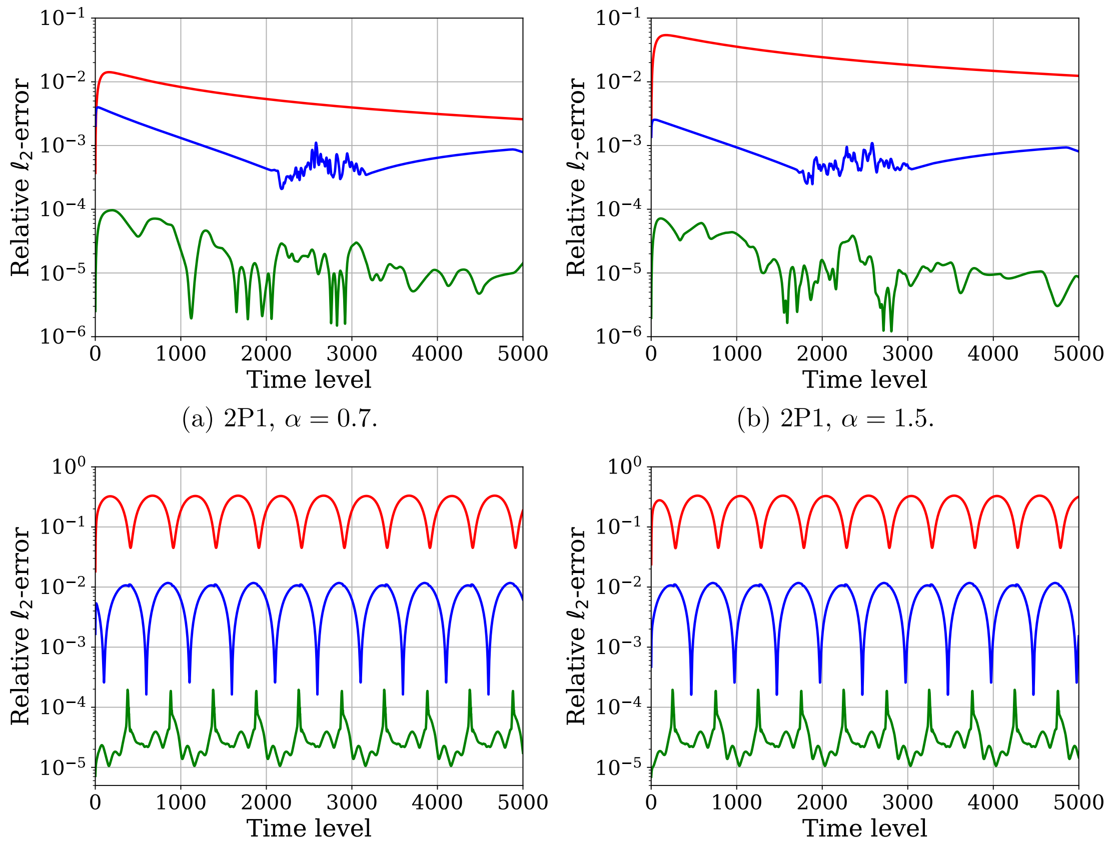
The temporal development of the models' predictive error is shown in Figure 4.23 for both Solution 2P1 and Solution 2P2 with $\alpha \in \{0.7, 1.5\}$. Figures 4.24 and 4.25 display the models' error fields at the final time level for Solution 2P1, while the final-time-level error fields for Solution 2P2 are displayed in Figures 4.26 and 4.27.

We first consider the PBM predictions for Solution 2P1 (cf. Figures 4.24 and 4.25). For this solution, the unknown source term $P = 1 - 2\alpha$ is negative, which means that it pulls heat out of the system. It is therefore not surprising that the PBM predictions are consistently too hot. Furthermore, since $|P|$ is greater for $\alpha = 1.5$ than for $\alpha = 0.7$, it is sensible that PBM performs worse for the former α-value than for the latter.

For Solution 2P2, the unknown P is spatially periodic, which causes the reference solution itself to be periodic as well. As the PBM is unaware of P and its effect on the temperature, it naturally predicts temperature fields that are too flat. That is, the PBM predictions are too cold in the hot regions and too hot in the cold regions (cf. the top right corners of Figures 4.26 and 4.27). Overall, the ℓ_2-error curves of Figure 4.23 show that PBM yields the lowest accuracy of all three models due to its inability to take the unknown P into account.

Moving over to DDM, we see from Figure 4.23 that DDM outperforms PBM in terms of accuracy in all four scenarios considered here. Indeed, DDM performs quite well, as the relative ℓ_2-error of its predictions stay at roughly 1% or less. However, the error fields in the lower left corner of Figures 4.24–4.27 show that DDM yields significant errors in some areas of the spatial domain. For Solution 2P1 (lower left corner of Figures 4.24 and 4.25), the DDM predictions are consistently too hot in the upper left corner of the domain and too cold in the domain's lower right corner. Comparing with the true solutions (upper left corners of the same figures), these errors are somewhat surprising, and it may look as though the DDM errors have simply been displayed with an inadvertent rotation or inversion. However, the author has double-checked that both figures do display the experimental data correctly. Hence, this observation is a good example of data-driven techniques obtaining errors which can be unpredictable and hard to interpret. Moving over to Solution 2P2, the DDM errors for this solution are somewhat easier to interpret; for $\alpha = 0.7$ (cf. Figure 4.26), the DDM prediction is generally too flat, while the DDM prediction for $\alpha = 1.5$ (cf. Figure 4.27) is generally too curved. This is consistent with the DNN having learnt the correct spatial periodicity of the true solution, but also having failed to learn the correct amplitude.

Finally, we consider the results of CoSTA-based HAM. As in most other experiments considered so far, HAM again demonstrates predictive accuracy superior to both PBM and DDM. The HAM errors for Solution 2P1, which are illustrated in the bottom right corners of Figures 4.24 and 4.25, are virtually unnoticeable with the color scheme that has been used. The HAM errors for Solution 2P2 are more significant, as they are clearly visible in the colder regions of the domain. However, these errors are still more than one order of magnitude lower than the PBM and DDM errors. Thus, it is clear that the hybrid model once again benefits from the use of the PBM to ease the learning task of its DNN. It is remarkable that this holds true despite the large errors of the stand-alone PBM, which periodically exceed 30% for Solution 2P2 (cf. bottom row of Figure 4.23).



(a) 2P1, $\alpha = 0.7$.

(b) 2P1, $\alpha = 1.5$.

(c) 2P2, $\alpha = 0.7$.

(d) 2P2, $\alpha = 1.5$.

Figure 4.23.: Solutions 2P1 and 2P2, interpolation: Relative ℓ_2-errors for $\alpha \in \{0.7, 1.5\}$ (— PBM, — DDM, — HAM).

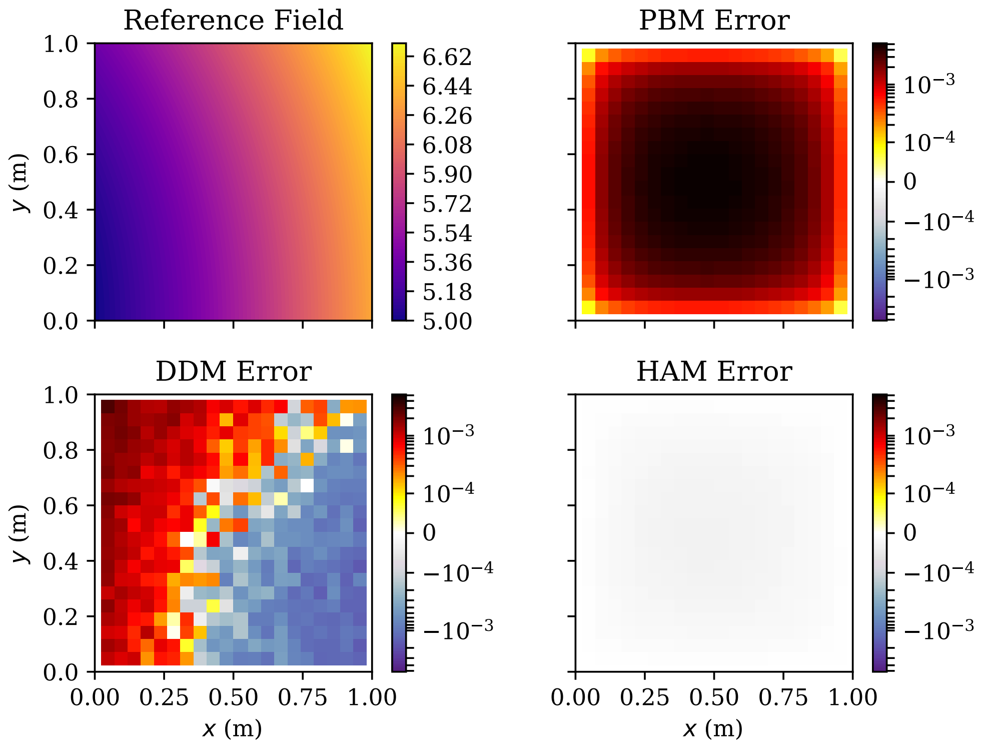


Figure 4.24.: Solution 2P1, $\alpha = 0.7$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

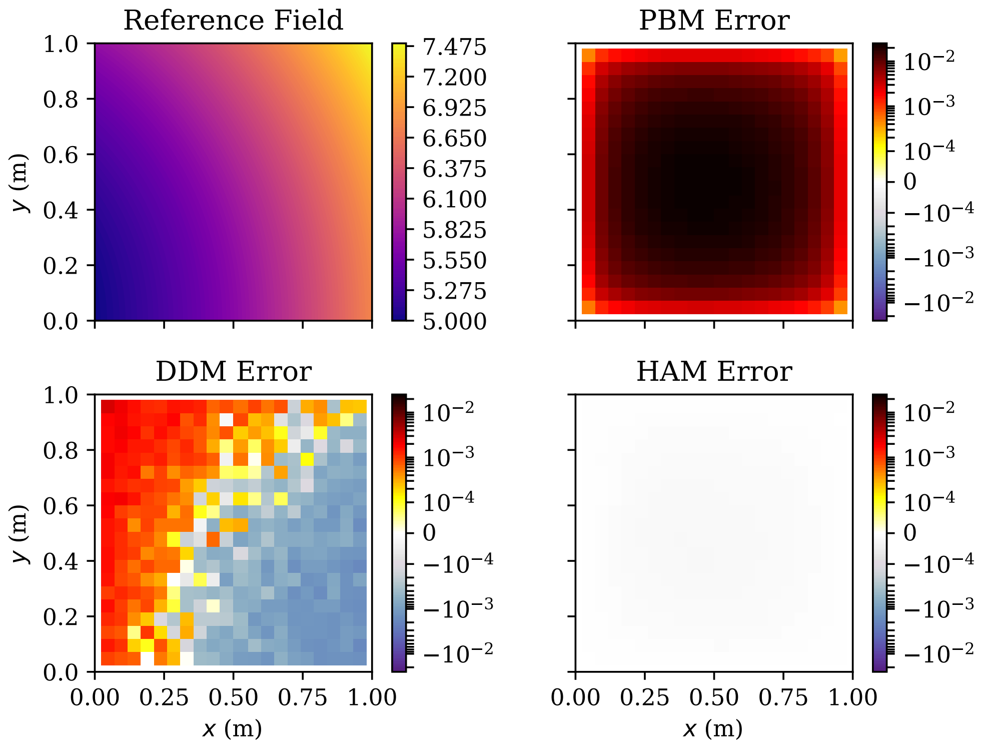


Figure 4.25.: Solution 2P1, $\alpha = 1.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

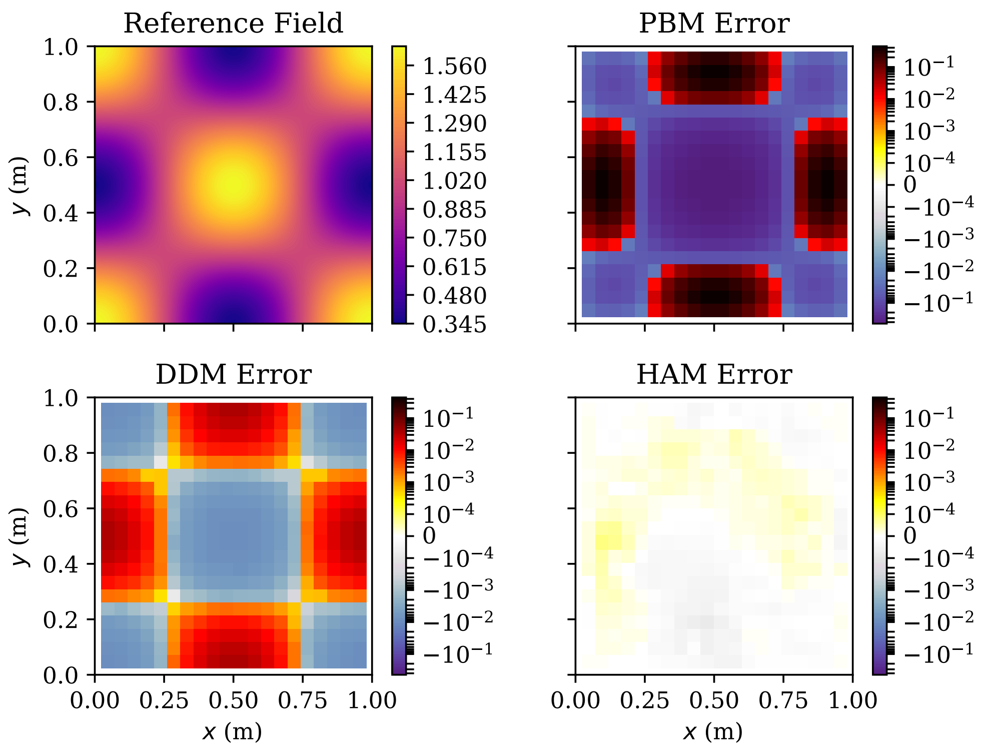


Figure 4.26.: Solution 2P2, $\alpha = 0.7$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

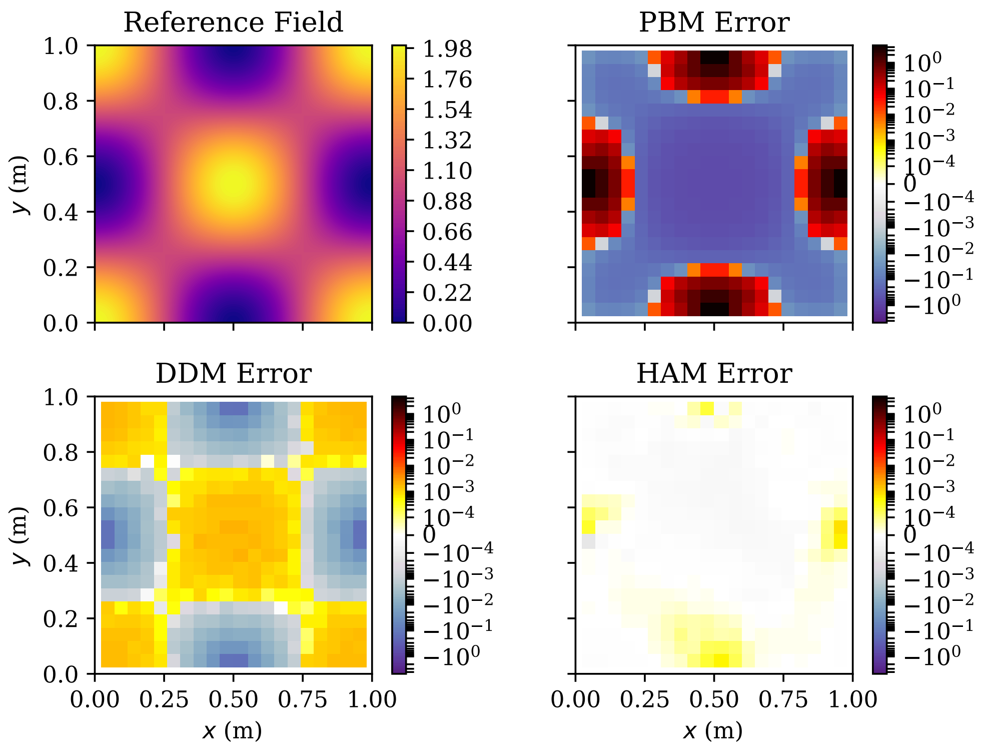


Figure 4.27.: Solution 2P2, $\alpha = 1.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

Extrapolation Scenarios

The results for the extrapolation scenarios are presented in Figure 4.28 (temporal development of ℓ_2-errors for Solutions 2P1 and 2P2), Figures 4.29 and 4.30 (error fields for Solutions 2P1), and Figures 4.31 and 4.32 (error fields for Solutions 2P2). These results tell much the same story as we have seen in the extrapolation scenarios of the 1D experiments. PBM generalizes well, but the modelling error caused by the unknown source term still results in PBM performing significantly worse than the other two models for three of the four scenarios considered (cf. Figure 4.28): Solution 2P1 with $\alpha = 2.5$ and Solution 2P2 with $\alpha \in \{-0.5, 2.5\}$. In these scenarios, HAM is by far the most accurate model, with DDM following thereafter and PBM being the least accurate model.

The results for Solution 2P1 with $\alpha = -0.5$ are somewhat different to those of the other scenarios. In terms of accuracy, the performance of DDM is roughly as good as that of PBM, while HAM outperforms the other two models by only a small margin, as can be seen from Figures 4.28a and 4.29. In this scenario, it is particularly interesting to compare the errors of the final-time-level predictions, as illustrated in Figure 4.29. We see that the PBM and HAM predictions (right column) are generally too cold, which is consistent with the unknown P being positive for this value of α. Furthermore, we observe that the HAM prediction is significantly warmer than the PBM prediction, which indicates that the DNN of the HAM model is able provide a correction that is qualitatively correct. However, the correction is too weak for the HAM model to achieve the same level of accuracy as for $\alpha \in \{0.7, 1.5, 2.5\}$. Nonetheless, this indicates that HAM generalizes better than DDM, which fails to realize that the hottest part of the domain in this scenario is the bottom right corner – not the top right corner, as for the other α-values.

We can also make an interesting observation from the DDM error field for Solution 2P1 with $\alpha = 2.5$, which is illustrated in the bottom left corner of Figure 4.30. This error field appears very noisy, as the DDM predicts temperatures which are too hot in some grid cells while simultaneously predicting temperatures that are too cold in neighbouring grid cells. A similar phenomenon can also be observed in the HAM error fields for Solution 2P2, as shown in the bottom right corner of Figures 4.31 and 4.32. In these figures, the noise is not as pronounced, but the correlation between the reference temperature fields and the error fields of the HAM predictions still appears to be rather weak. That is to say, the HAM predictions also seem to contain some level of noise.

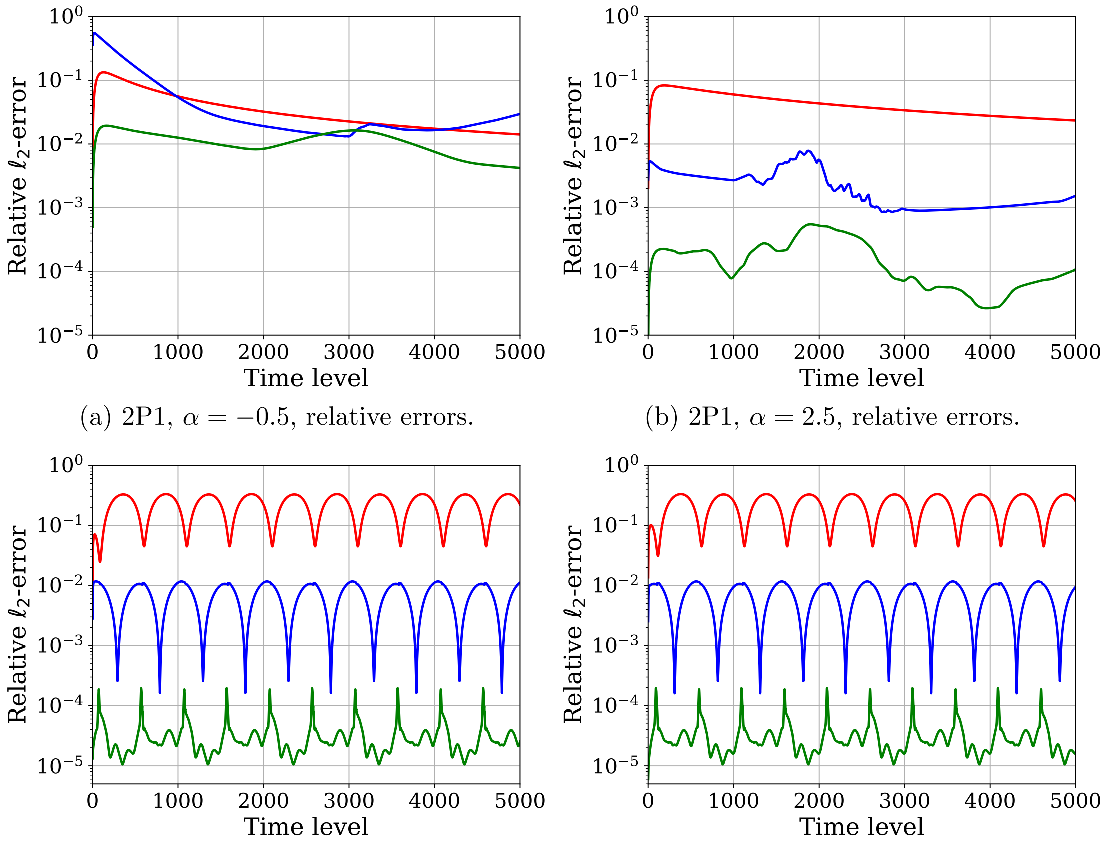
The perceived noisiness of the HAM and DDM predictions can be attributed to the DNN training process, which does not encourage the DNN outputs to be smooth in space. As discussed in Section 4.1, the DNNs are trained using the mean square error cost functions, which does not take into account the spatial distribution of the DNNs' errors. Thus, this cost function does not reward smooth DNN outputs over irregular outputs, which means that the DNNs have no reason to prefer smooth outputs over irregular outputs. As irregular outputs have larger entropy (more disorder) than smooth outputs, it is then much more likely that a DNN will end up producing irregular outputs than smooth outputs.[13] As discussed in Section 4.2.3, small error fluctuations in an otherwise correct prediction may amplify over time, possible even causing long-term divergent behaviour. Therefore, further research into how appropriate smoothness can be imposed on the output of DNNs used in CoSTA-based HAM models would be highly relevant for the future development of CoSTA. One possibility is to add an extra term to the cost function used during DNN training, in order to enforce a smoothness requirement. This

[13]This follows intuitively from the realization that there exists many more possibilities for a system to be disordered than for it to be ordered.

term could for instance be proportional to the sum of absolute differences between the DNN's prediction and the corresponding target output at neighbouring grid cells. One possible mathematical formulation of such a cost term is

$$\frac{1}{N_O - 1} \sum_{l=1}^{N_O - 1} \big| |O_l - (O_{\text{ref}})_l| - |O_{l+1} - (O_{\text{ref}})_{l+1}| \big|, \tag{4.7}$$

where we have used the same notation as in Equation (4.6). Enforcing smoothness by minimizing the finite difference approximations of higher-order spatial derivatives is also a possibility. Moreover, one may consider applying some post-processing like Gaussian blur to the DNN output. However, for any of these techniques to work as intended, one needs *a priori* knowledge of what the smallest length-scale of the problem is. otherwise, one could end up smoothing away variations of physical significance. Another issue is that there exists problems where the true corrective source term itself is far from smooth. An example of such a problem is considered in Appendix A. For such problems, naive smoothening of the DNN output would be detrimental to the accuracy of the model.



(a) 2P1, $\alpha = -0.5$, relative errors.

(b) 2P1, $\alpha = 2.5$, relative errors.

(c) 2P2, $\alpha = -0.5$, relative errors.

(d) 2P2, $\alpha = 2.5$, relative errors.

Figure 4.28.: Solutions 2P1 and 2P2, extrapolation: Relative ℓ_2-errors for $\alpha \in \{0.7, 1.5\}$ (— PBM, — DDM, — HAM).

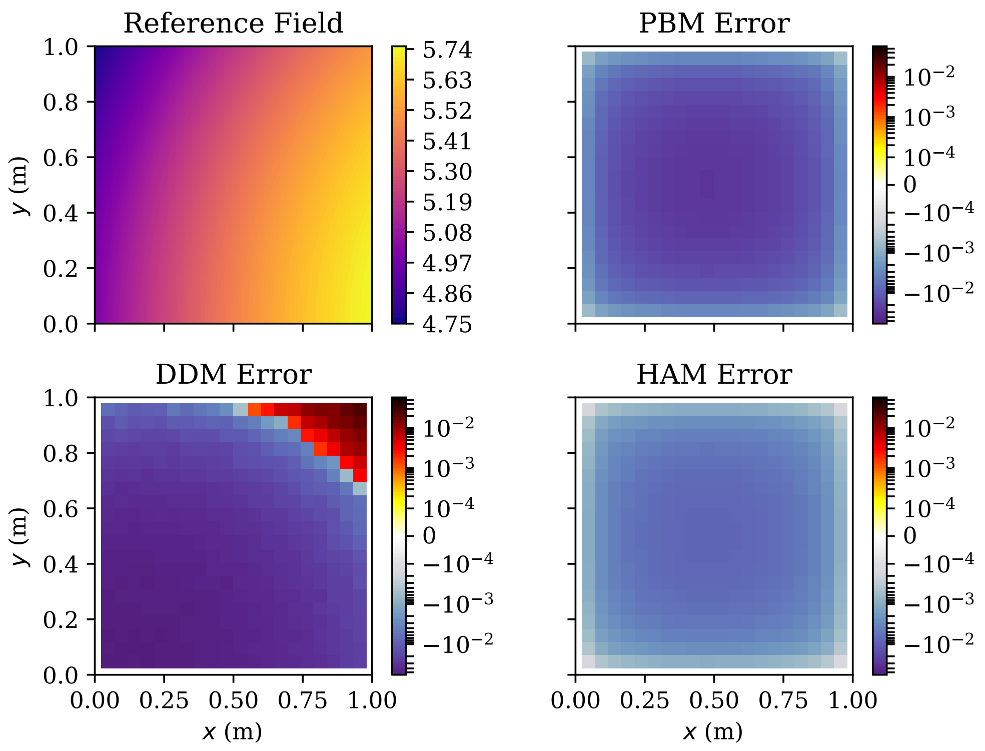


Figure 4.29.: Solution 2P1, $\alpha = -0.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

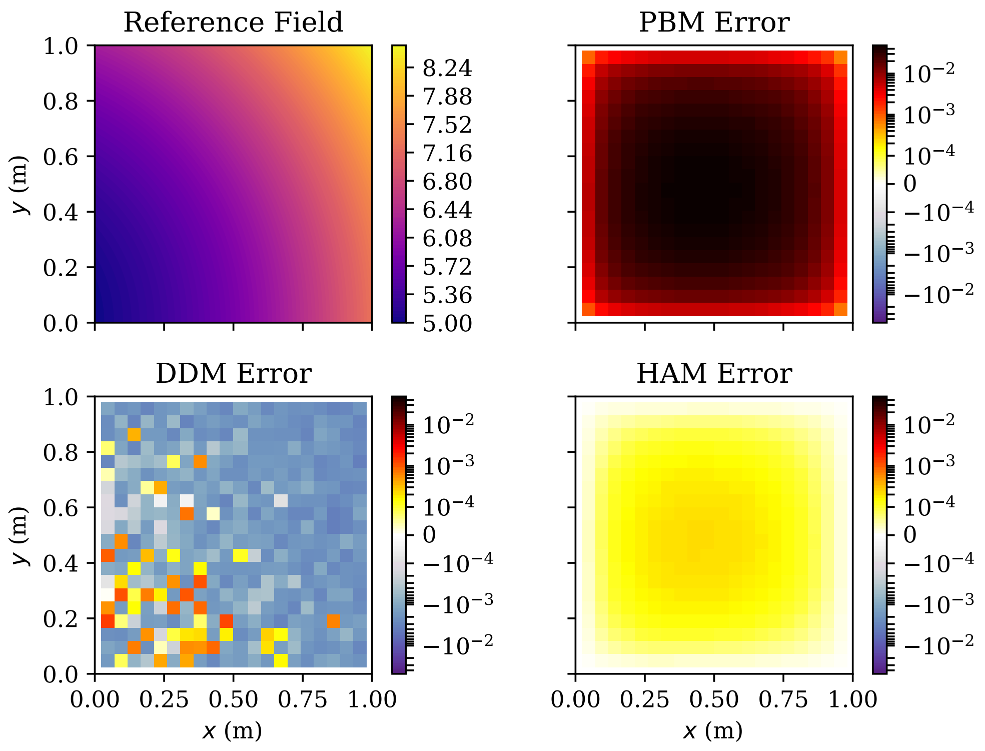


Figure 4.30.: Solution 2P1, $\alpha = 2.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

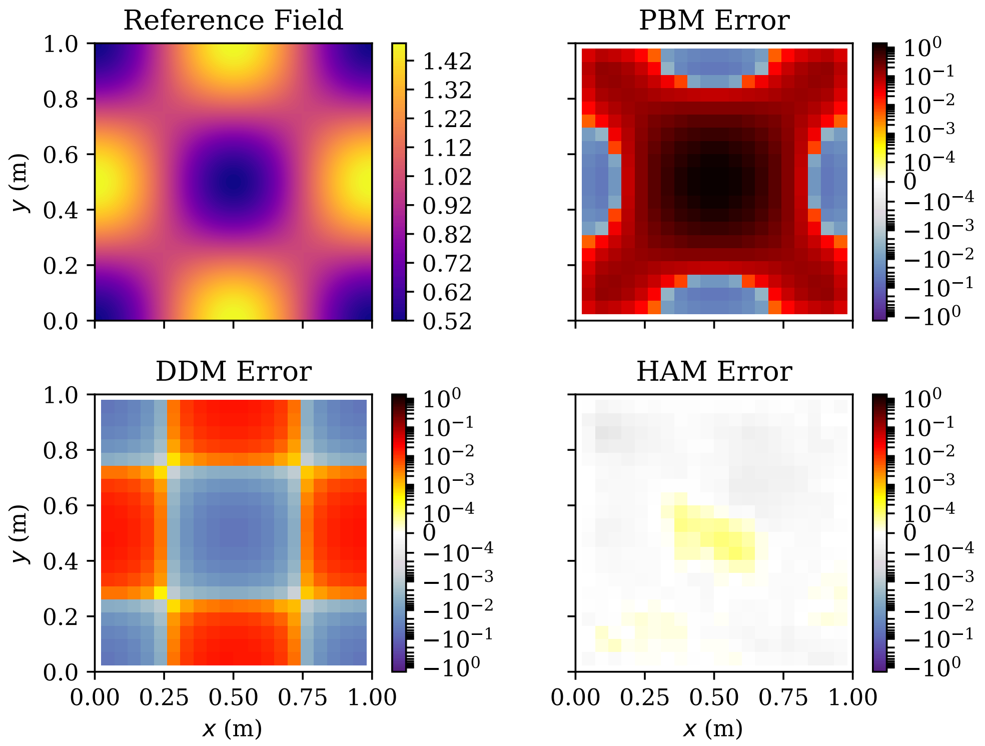


Figure 4.31.: Solution 2P2, $\alpha = -0.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

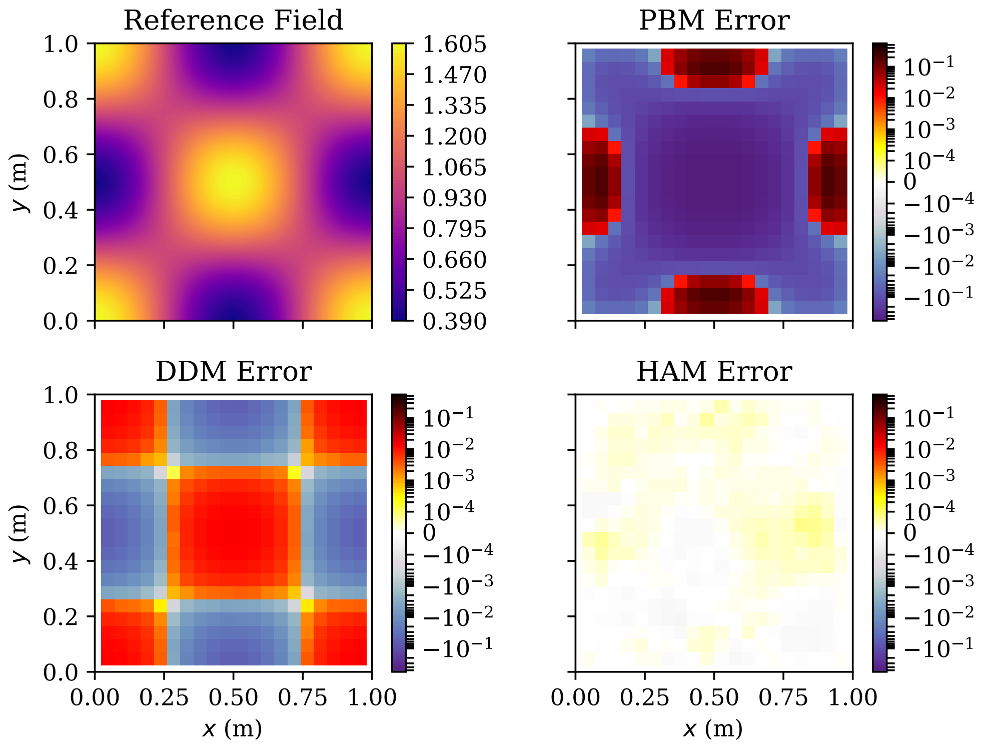


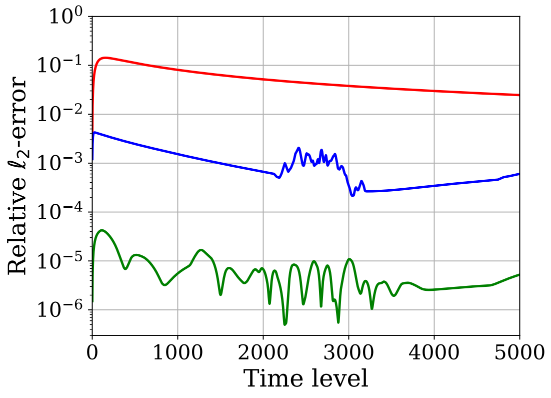
Figure 4.32.: Solution 2P2, $\alpha = 2.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

4.3.2. Experiments with Unknown Conductivity

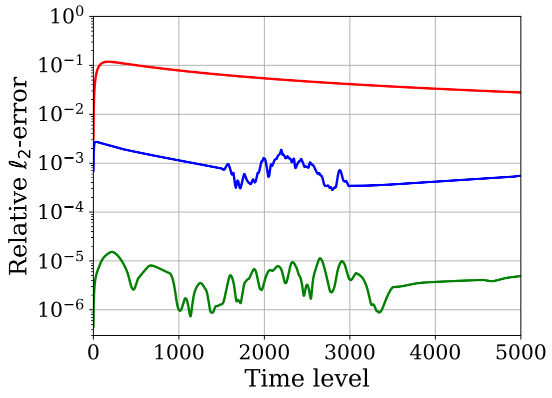
In this second and final series of 2D experiments, we consider two experiments where the conductivity k is unknown, while the source term P is known. This means we set $k = 1$ when defining $\mathbb{A}$ and $\boldsymbol{b}$ in Equations (4.1), (4.3) and (4.5), just like we did for the experiments of Section 4.2.3. For the experiments considered here, we use Solutions 2k1 and 2k2 (cf. Table 4.2) with $N_j = N_i = 20$, which yields 400 grid cells. As the results of these experiments do not warrant any lengthy discussions, we discuss both the interpolation and extrapolation scenarios at once in the paragraphs below. The results from the interpolation scenarios are displayed in Figures 4.33–4.37, while the extrapolation scenario results can be found in Figures 4.38–4.42.

The ℓ_2-error curves for the interpolation scenarios $\alpha \in \{0.7, 1.5\}$ and the extrapolation scenarios $\alpha \in \{-0.5, 2.5\}$ are shown in Figures 4.33 and 4.38, respectively. From these figures, we see that HAM is generally more accurate than DDM, which in turn is generally more accurate than PBM. The exceptions are the extrapolation scenarios for Solution 2k1. For Solution 2k1 with $\alpha = -0.5$, we see that HAM is the most accurate model, while DDM fails completely to generalize to this case and performs the worst of all three methods (cf. Figures 4.38a and 4.39). The DDM generalizes much better to Solution 2k1 with $\alpha = 2.5$, and actually matches the accuracy of HAM in this scenario. However, while HAM and DDM achieve the same accuracy in this scenario, as measured by the ℓ_2-norm, there is still an important difference between their predicted solutions. From Figure 4.40, we see that HAM's error field (bottom right) is completely smooth, indicating that the corrective source term resulted in a correction that was qualitatively correct, but too weak. In comparison, the DDM's error field (bottom left) is less smooth, and has a peak in the upper left corner of the spatial domain which has no obvious relation to the reference temperature field (top left). Thus, the HAM prediction is easier to interpret than that of the DDM, and therefore also more trustworthy.

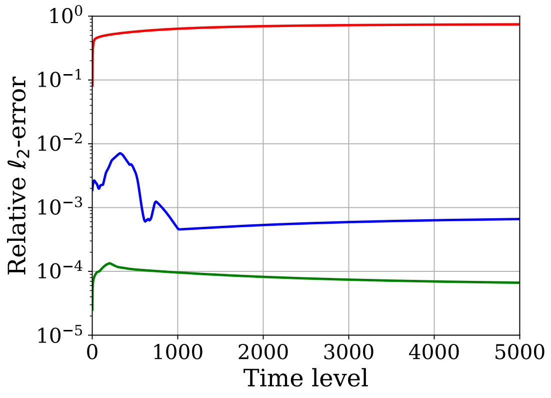
Another observation worth noting is that we once again see noise-like fluctuations in the error fields of HAM and DDM. These fluctuations are particularly noticeable in Figures 4.34 and 4.35 (for DDM) and Figures 4.41 and 4.42 (for both HAM and DDM). At first glance, it may appear from Figures 4.36 and 4.37 that the PBM predictions are affected by some random noise as well. However, this is not the case. Upon close inspection, one case see that the PBM predictions are point-symmetric with respect to the center of the spatial domain. The complex pattern is caused by the source term P, which itself is a complex function of x and y. In the reference solution, some of this complexity is cancelled out by the non-constant conductivity. However, the complexity of P is retained in the PBM predictions, since the PBM assumes the conductivity to be constant. The only non-deterministic aspect in the PBM is rounding error. Since the Implicit Euler FVM is a stable numerical solver, it never amplifies rounding errors across time levels, and rounding errors are therefore not observable in the figures presented here.



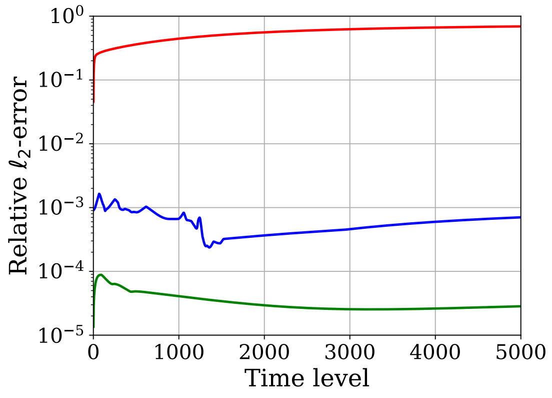
(a) 2k1, $\alpha = 0.7$, relative errors.



(b) 2k1, $\alpha = 1.5$, relative errors.



(c) 2k2, $\alpha = 0.7$, relative errors.



(d) 2k2, $\alpha = 1.5$, relative errors.

Figure 4.33.: Solutions 2k1 and 2k2, interpolation: Relative ℓ_2-errors for $\alpha \in \{0.7, 1.5\}$ (— PBM, — DDM, — HAM).

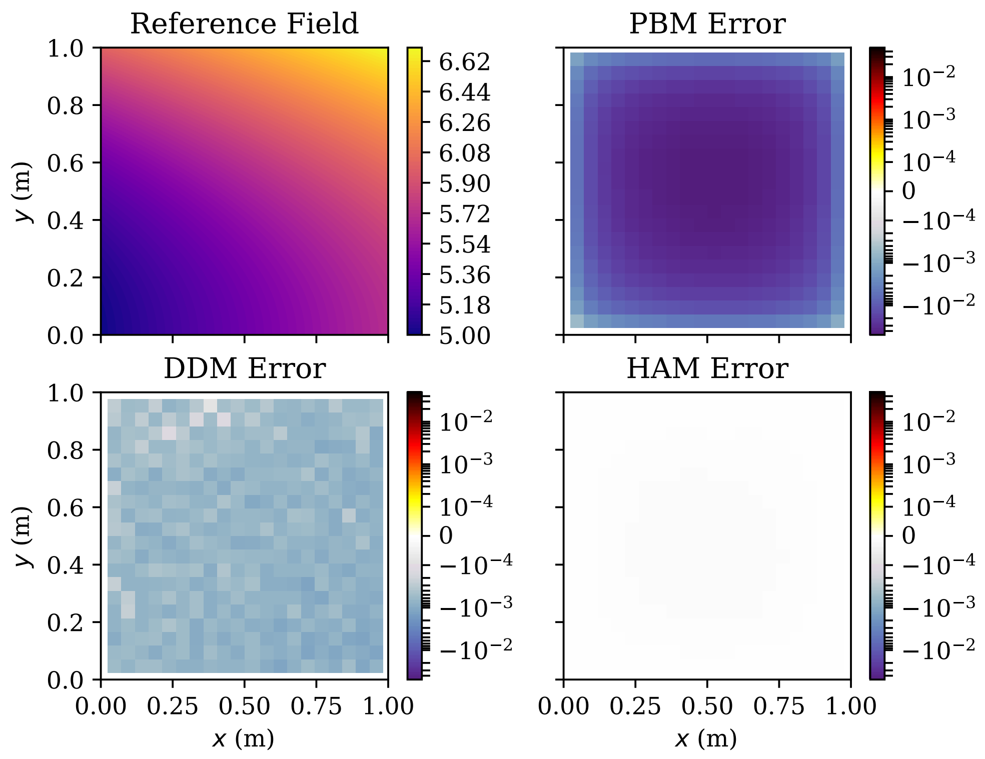


Figure 4.34.: Solution 2k1, $\alpha = 0.7$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

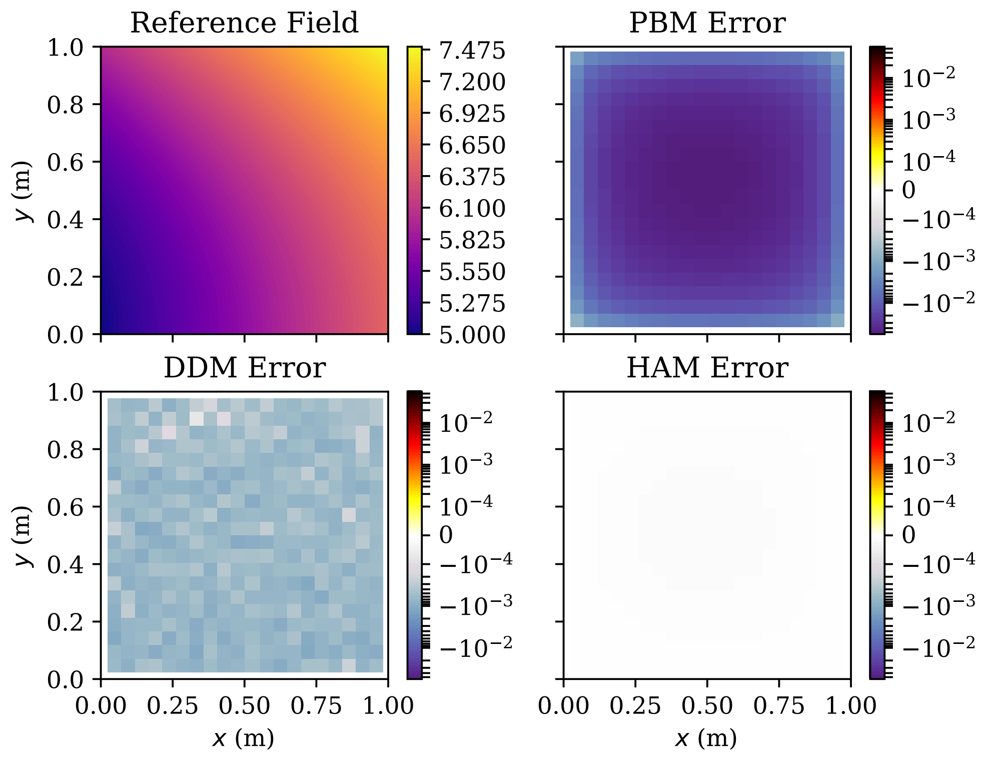


Figure 4.35.: Solution 2k1, $\alpha = 1.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

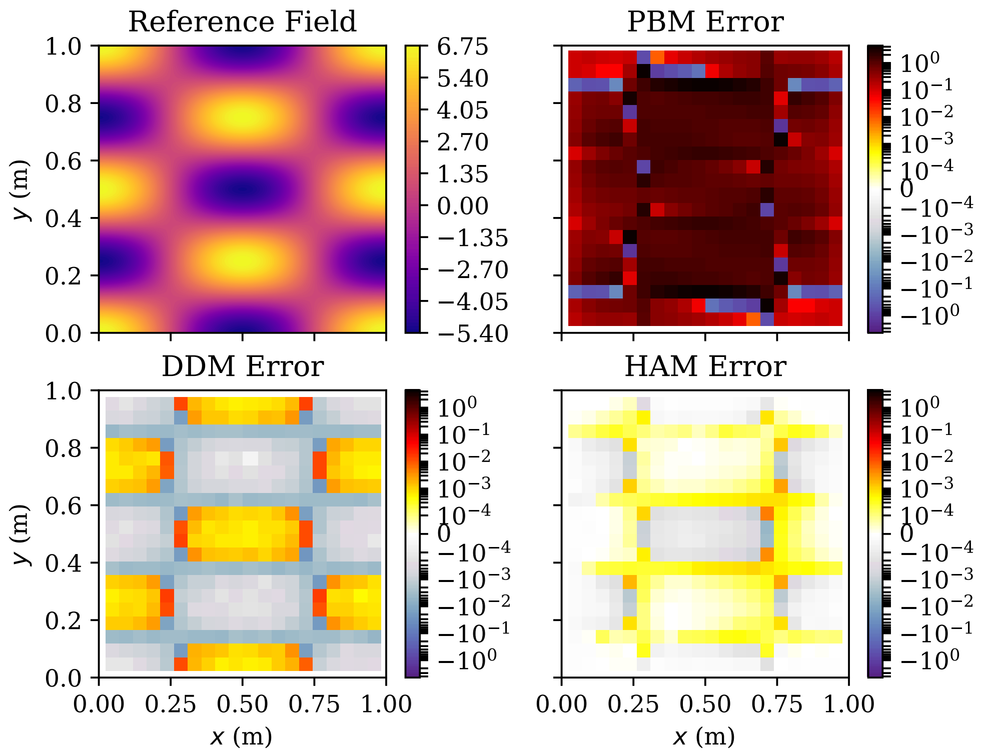


Figure 4.36.: Solution 2k2, $\alpha = 0.7$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

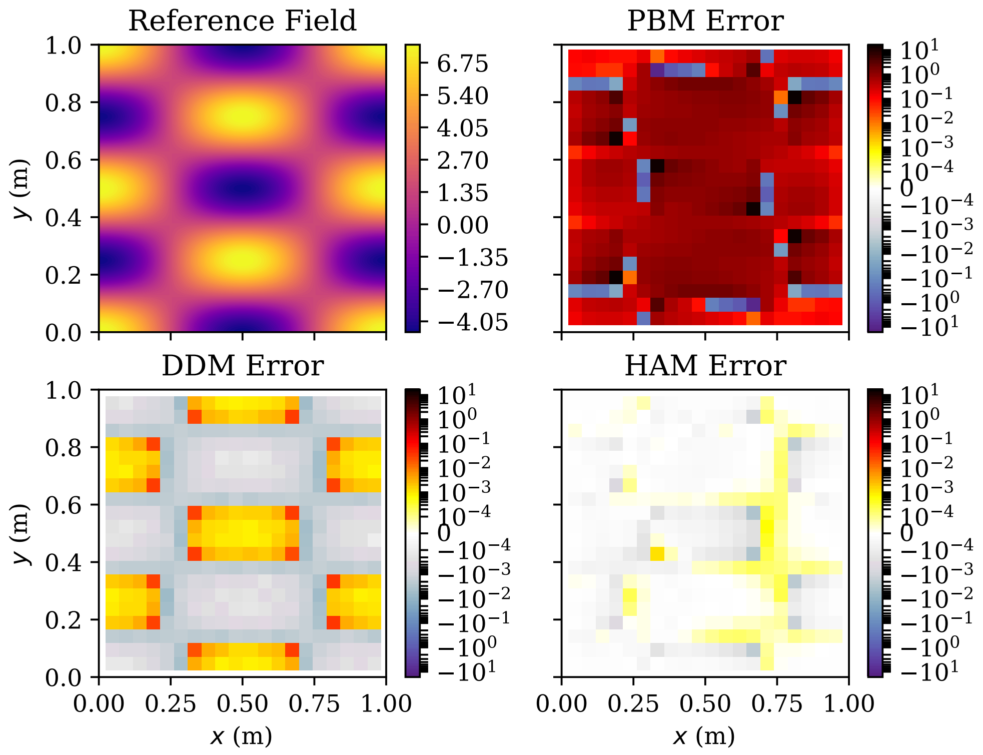
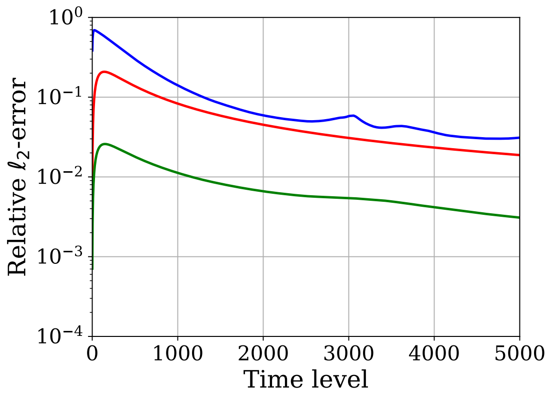
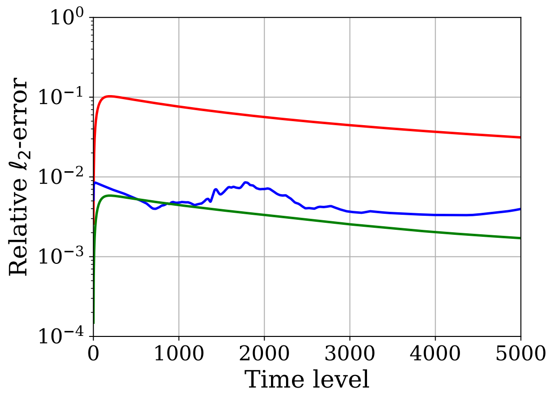


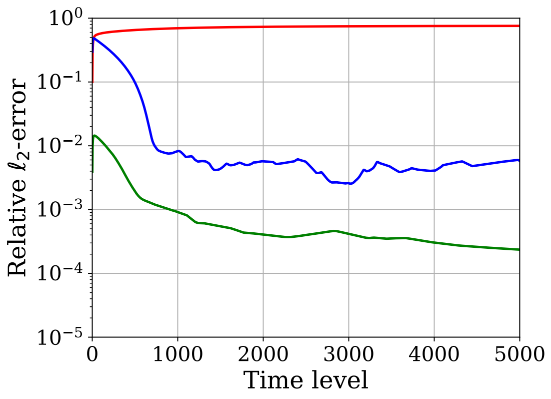
Figure 4.37.: Solution 2k2, $\alpha = 1.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.



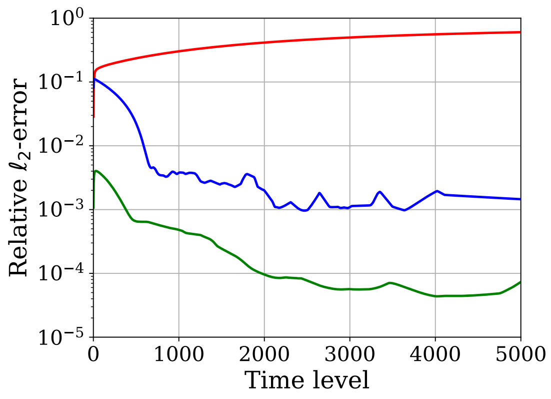
(a) 2k1, $\alpha = -0.5$, relative errors.



(b) 2k1, $\alpha = 2.5$, relative errors.



(c) 2k2, $\alpha = -0.5$, relative errors.



(d) 2k2, $\alpha = 2.5$, relative errors.

Figure 4.38.: Solutions 2k1 and 2k2, extrapolation: Relative ℓ_2-errors for $\alpha \in \{0.7, 1.5\}$ (— PBM, — DDM, — HAM).

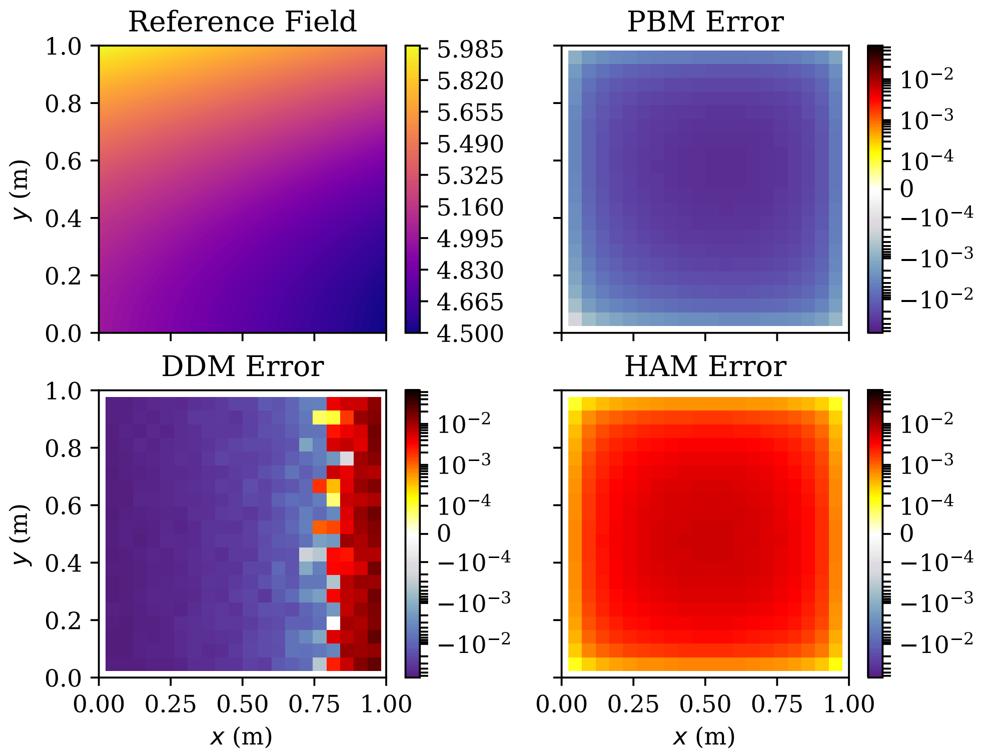


Figure 4.39.: Solution 2k1, $\alpha = -0.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

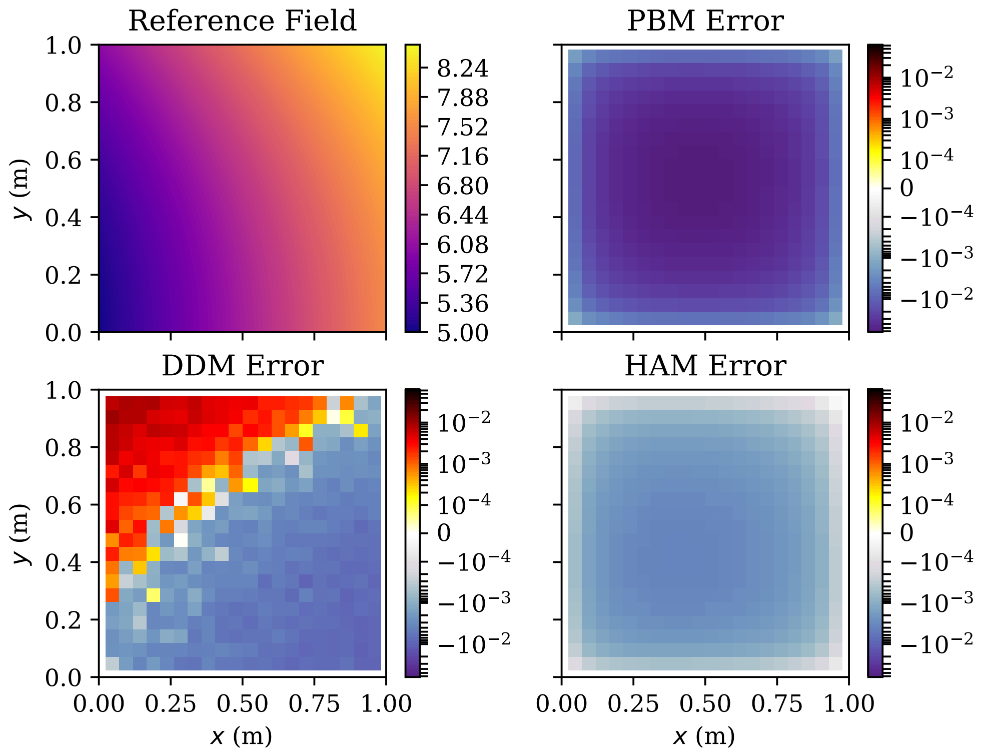


Figure 4.40.: Solution 2k1, $\alpha = 2.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

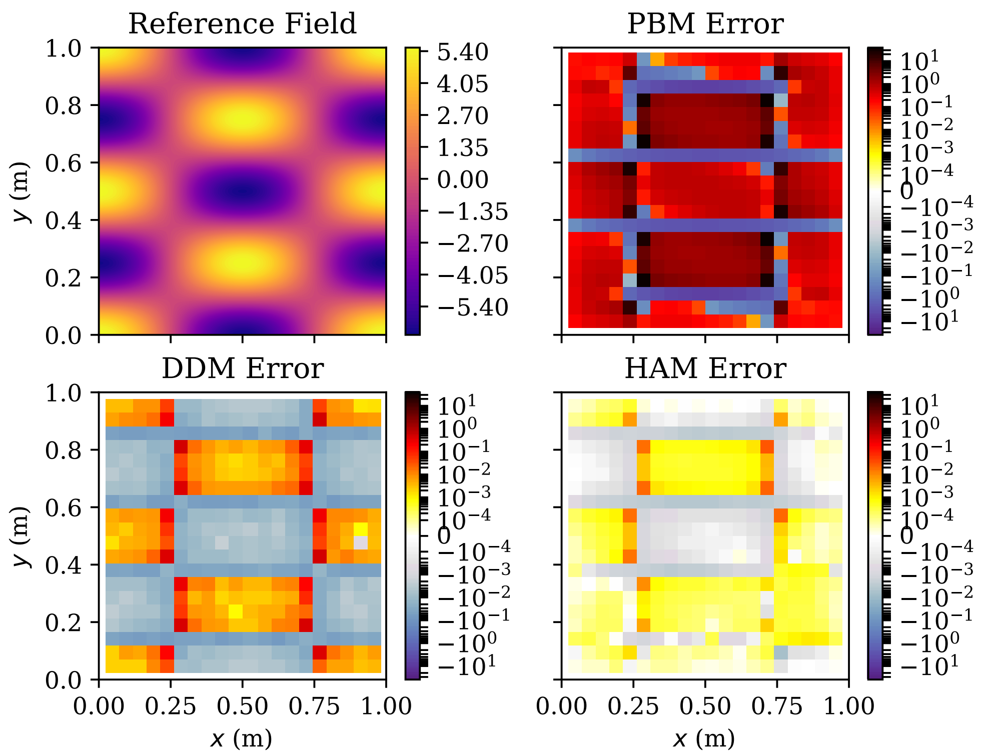


Figure 4.41.: Solution 2k2, $\alpha = -0.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

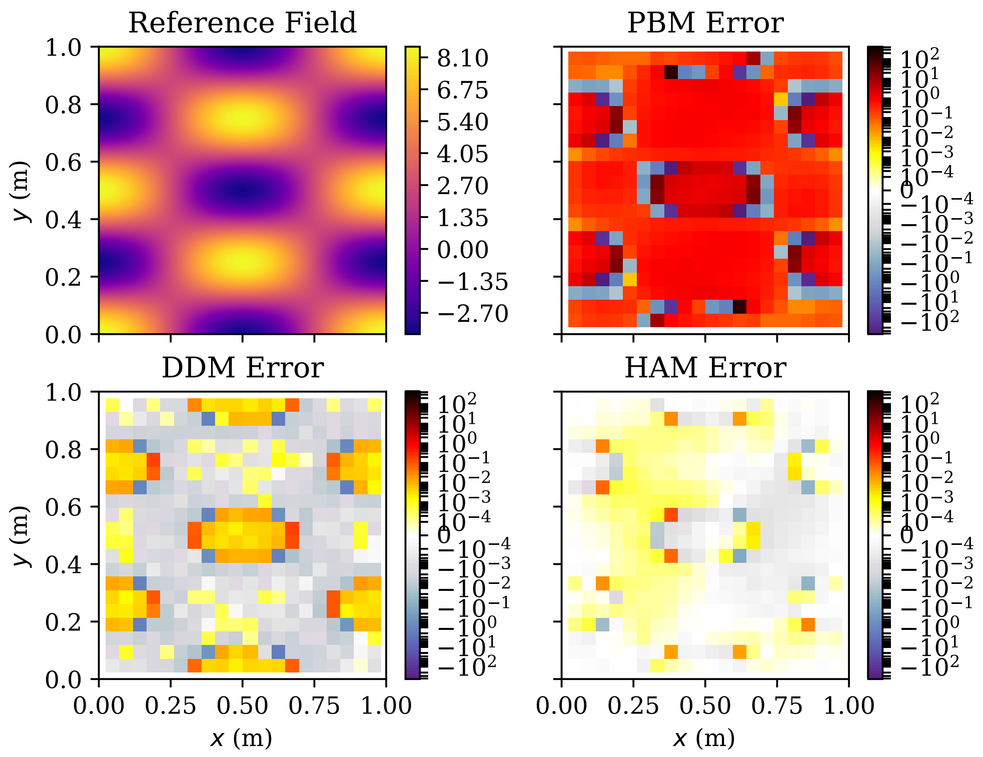


Figure 4.42.: Solution 2k2, $\alpha = 2.5$: Reference temperature field and relative ℓ_2-errors of PBM, DDM and HAM.

4.3.3. Summary of Experiments in Two Dimensions

At large, the experiments considered in this section indicate that CoSTA-based HAM is more accurate than PBM and DDM in 2D heat transfer problems. This supports the findings from the 1D experiments of the previous section, where we found that combining a PBM and a DDM in a single hybrid model yields better accuracy than using either of the two traditional modelling approaches on their own. The experimental results of the present section also indicate that HAM generalizes significantly better than DDM in both interpolation and extrapolation scenarios. In fact, HAM generalizes well enough the be the most accurate model even in extrapolation scenarios that are qualitatively different to the scenarios seen during training.

On the downside, the experiments of this section has revealed that both DDM and HAM are prone to creating noisy temperature predictions. As the influence of noise on the long-term behaviour of both HAM (and also DDM) is poorly understood at present (cf. Section 4.2.4), this represent a potential trustworthiness issue which should be addressed in future work. However, it is worth mentioning that HAM is significantly less noise-prone than DDM in the experiments considered in this section.

4.4. Predictive Uncertainty

As discussed in Section 2.3.3, DNNs are customarily trained using stochastic learning algorithms. A result of this is that different instances of the same DNN may yield different predictions when presented with exactly the same data. Consequently, predictions made by a model with one or more DNN-based components are associated with some inherent uncertainty. This negatively affects the model's explainability, and thereby also its trustworthiness. In this section, we investigate the inherent uncertainty of the DDM and HAM models used for the numerical experiments discussed in Section 4.2.2.[14] This investigation is motivated by our third research question, which concerns the trustworthiness of CoSTA-based HAM models (cf. Section 1.2).

In our simple study of model variability, we repeat the experiments of Section 4.2.2 with a small modification: Instead of training one DDM model instance and one HAM model instance per manufactured solution, we now train five separate instances of each model for each manufactured solution. Each of these model instances is initialized using the standard PyTorch initialization scheme, but with its own unique seed. For each group of five model instances, we calculate the component-wise mean and standard deviation of their predicted temperature profiles at the final time level. We label these statistical quantities $\mu[\boldsymbol{T}_{\mathrm{d}}^{N_t-1}]$ and $\mathrm{std}[\boldsymbol{T}_{\mathrm{d}}^{N_t-1}]$ for DDM, and $\mu[\boldsymbol{T}_{\mathrm{h}}^{N_t-1}]$ and $\mathrm{std}[\boldsymbol{T}_{\mathrm{h}}^{N_t-1}]$ for HAM. While the means are only used for visualization purposes, we use the empirical standard deviations as heuristic measures of the models' inherent uncertainty. This approach to uncertainty estimation is used e.g. in the work by Pawar et al. (2021b) on physics-guided machine learning. It is a far simpler approach than the Bayesian NNs that are commonly used for high-quality predictive uncertainty estimation, because it does not require any alterations to our chosen DNN architectures or training procedures.

The empirical means and standard deviations for our DDM and HAM models were used to generate Figures 4.43 and 4.44. These figures illustrate the predictive uncertainty of the models for Solutions P1 and P3, respectively. In each figure, the blue-coloured area is defined as the area between the two curves

$$\mu[\boldsymbol{T}_{\mathrm{d}}^{N_t-1}] + 5\,\mathrm{std}[\boldsymbol{T}_{\mathrm{d}}^{N_t-1}] \quad \text{and} \quad \mu[\boldsymbol{T}_{\mathrm{d}}^{N_t-1}] - 5\,\mathrm{std}[\boldsymbol{T}_{\mathrm{d}}^{N_t-1}], \tag{4.8}$$

[14]We do not discuss the PBM here, since it is deterministic up to machine epsilon.

and it is taken to represent the predictive uncertainty of the DDM model for the corresponding solution. Analogously, the area between the two curves

$$\mu[\boldsymbol{T}_{\mathrm{h}}^{N_t-1}] + 5\,\mathrm{std}[\boldsymbol{T}_{\mathrm{h}}^{N_t-1}] \quad \text{and} \quad \mu[\boldsymbol{T}_{\mathrm{h}}^{N_t-1}] - 5\,\mathrm{std}[\boldsymbol{T}_{\mathrm{h}}^{N_t-1}], \tag{4.9}$$

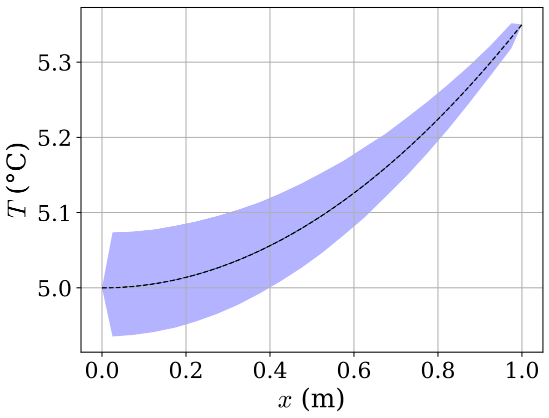
which is coloured in green, is taken to represent the predictive uncertainty of the HAM model. Where the two areas overlap, the result is a dark shade of turquoise. With these definitions, it is clear that a large coloured area corresponds to a large empirical standard deviation. As stated above, we assume that a large empirical standard deviation correlates to a large model uncertainty.[15]

From Figures 4.43 and 4.44, we see that the DDM model has a significantly larger empirical standard deviation than the HAM model in all scenarios except for Solution P3 with $\alpha = -0.5$. However, as is clear from Figure 4.44c, neither DDM nor HAM provide qualitatively correct predictions for that scenario. Thus, that result merely indicates that DDM is more certain of its wrong prediction than is HAM, which is arguably not a favourable trait of the DDM. In all other scenarios (where HAM does provide accurate predictions (cf. Section 4.2.2), such that low uncertainty is indeed desirable) the standard deviation of HAM is far smaller than that of DDM.

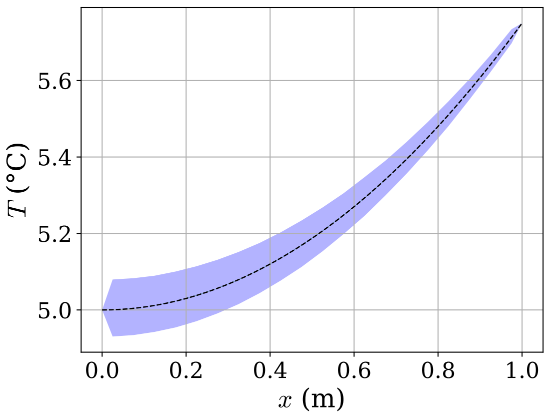
In addition to Figures 4.43 and 4.44, which correspond to Solutions P1 and P3, we have also created similar figures for Solutions P2 and P4, and they can be found in Appendix E. These figures are not included here simply because the green areas serving as a proxy for the HAM uncertainty are so small that they are not visible. As the blue areas corresponding to the DDM uncertainty are clearly visible in these same figures, our results indicate that the HAM model has less inherent uncertainty than the DDM also for Solutions P2 and P4. Overall, our experiments indicate that the model uncertainty of DDM is larger than that of HAM by one order of magnitude or more for all four Solutions P1–P4. This makes the CoSTA-based HAM model more trustworthy than the DDM.

While the results presented in this section are very promising for CoSTA-based HAM, we highlight that they should be viewed with some caution. In the supplementary material of their work on uncertainty estimation using deep ensembles, Lakshminarayanan et al. (2017) demonstrate that the heuristic we have been using here is not a well-calibrated uncertainty estimate. Therefore, we cannot rule out the possibility that the results presented here are incidentally biased in favour of HAM. A more thorough investigation of model uncertainty is therefore warranted. Such an investigation can e.g. be conducted in a Bayesian framework, or using the deep ensemble approach advocated by Lakshminarayanan et al. (2017).

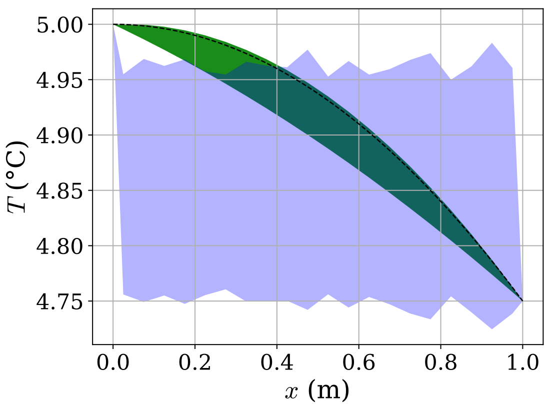
[15]The validity of this assumption will be discussed at the end of the section.



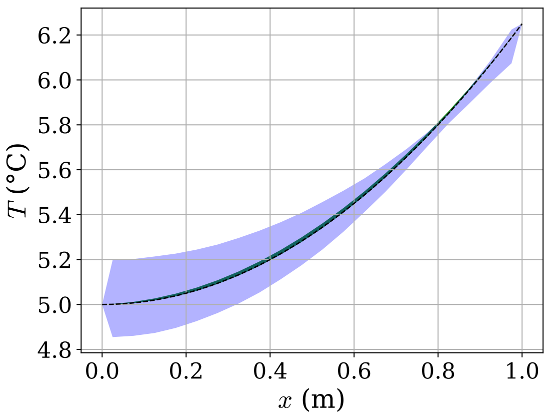
(a) $\alpha = 0.7$.



(b) $\alpha = 1.5$.

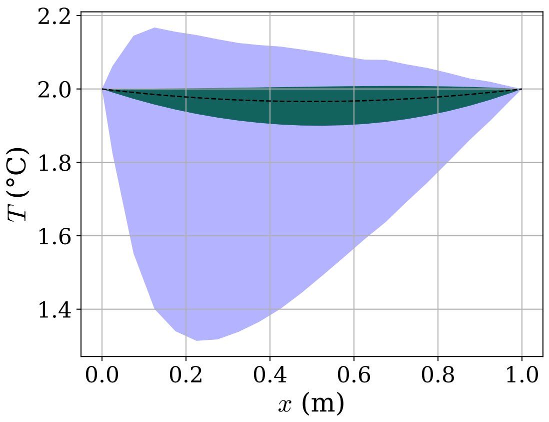


(c) $\alpha = -0.5$.

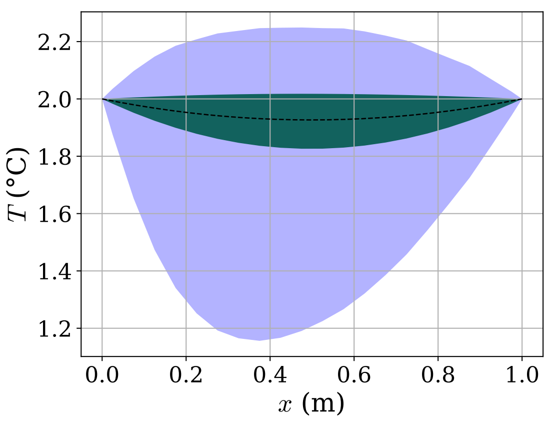


(d) $\alpha = 2.5$.

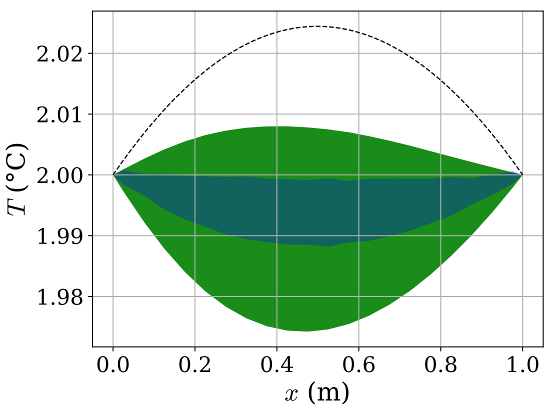
Figure 4.43.: Solution P1: Visualization of the inherent uncertainty of DDM and HAM models, where a large coloured area corresponds to a large model uncertainty. Blue area corresponds to uncertainty of DDM, and green area corresponds to uncertainty of HAM. Where the two areas overlap, the result is a dark shade of turquoise. The dashed line represents the exact solution P1.



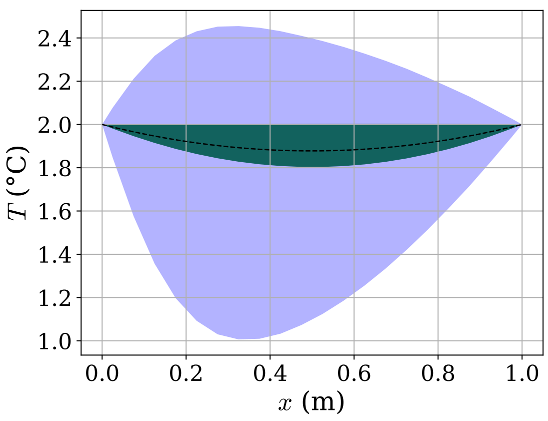
(a) $\alpha = 0.7$.



(b) $\alpha = 1.5$.



(c) $\alpha = -0.5$.



(d) $\alpha = 2.5$.

Figure 4.44.: Solution P3: Visualization of the inherent uncertainty of DDM and HAM models, where a large coloured area corresponds to a large model uncertainty. Blue area corresponds to uncertainty of DDM, and green area corresponds to uncertainty of HAM. Where the two areas overlap, the result is a dark shade of turquoise. The dashed line represents the exact solution P3.

4.5. Grid Refinement Studies

Grid refinement studies are commonly used for verifying the implementation of numerical solvers such as the Implicit Euler FVM described in Section 2.2.3.[16] These studies are therefore widely seen in conjunction with physics-based modelling. However, to the author's knowledge, little research has been performed regarding what can be learnt from grid refinement on purely data-driven or hybrid models. In this section, we conduct a series of novel grid refinement studies where we include both DDM and HAM models, in addition to the customary PBM. The structure of the section is as follows: First, we give a brief introduction to the basics of grid refinement studies. We then explain how the grid refinement studies of the present work were conducted. At last, we present and discuss the results of these grid refinement studies.

4.5.1. Background

As the name suggests, grid refinement studies are studies where we investigate the effect of refining the discretization (grid) used by some predictive model. When performing grid refinement studies in the context of PBMs, one typically assumes that the true governing equation is known exactly, such that the discretization used in the numerical solver is the PBM's only source of error. For any *convergent* numerical solver, the discretization error goes to zero in the limit of an "infinitely fine" discretization, by definition. For 1D time-dependent problems such as those we consider in this section, an "infinitely fine" discretization corresponds to the limits $\Delta t, \Delta x \to 0$, or equivalently $N_t, N_j \to \infty$. The speed at which the error goes to zero as the (spatial or temporal) grid is refined, is known as the solver's *order of accuracy.* LeVeque (2002) defines a method's (spatial)[17] order of accuracy, λ, using the following equation:[18]

$$\|\mathcal{E}\| = C(\Delta x)^{\lambda} \ + \ \text{higher order terms in } \Delta x \tag{4.10}$$

Here, $\mathcal{E}$ is the error of the numerical solution with respect to the true solution at some time t, $\|\cdot\|$ is some arbitrary norm and C is some constant which depends on the choice of norm $\|\cdot\|$, the temporal discretization, the time t and the solution being modelled. If $\lambda \in \mathbb{N}$, we say that the numerical solver is λth order accurate. Tannehill et al. (1997, page 48) show that the central difference approximations used in our derivation of the Implicit Euler FVM are second order accurate, which means that the Implicit Euler FVM itself is also second order accurate (in space).

If the order of accuracy λ for some numerical solver is known, then grid refinement studies can be used to verify any implementation of that solver. A general procedure for conducting grid refinement studies is described in Algorithm 1. The crux of the procedure is to calculate the solver's empirical order of accuracy λ_ν (see formula in Algorithm 1) at different grid refinement levels $\nu = 1, \dots, N_\nu$, where N_ν is the number of grid refinements performed. If the solver has been correctly implemented (and, for time-dependent problems, if the temporal discretization error is sufficiently small), we

[16]See e.g. the online resource by Slater (2021) and the textbook by Roache (1998) cited therein.

[17]A temporal order of accuracy can be defined analogously, but we do not consider temporal grid refinement studies in this work, so its definition is not given here.

[18]For time-dependent problems, $\mathcal{E}$ also has terms which are proportional to Δt. These are not included in LeVeque's definition, but it is intuitively obvious that they must exist as a temporal discretization will not generally be error-free. However, in spatial grid refinement studies, the contribution of the temporal discretization to the total error is assumed negligible. Some care must be taken when choosing Δt to ensure that this assumption actually holds in practice.

will observe $\lambda \approx \lambda_\nu \ \forall \ \nu \in \{1, \ldots, N_\nu\}$. An alternative to calculating λ_ν is to create a plot of data points $(N_j, \mathcal{E}(N_j))$. According to Equation (4.10), these data points should (approximately) lie on a line with slope $-\lambda$ in a loglog-plot.

Algorithm 1: General procedure for verifying the correct implementation of a PBM without modelling error using a grid refinement study. We use the formal notation $u_\mathrm{p} = \mathrm{PBM}(N_j)$ to denote the PBMs approximation of u when a spatial discretization with N_j grid cells is used.

```
Choose some initial number of grid cells $N_j \in \mathbb{N}$.
Choose some grid refinement factor $\zeta \in \mathbb{N}$.
Choose some number $N_\nu \in \mathbb{N}$ of grid refinements to perform.
**for** $\nu$ in $\{0, \ldots, N_\nu\}$ **do**
    $u_\mathrm{p} \leftarrow \mathrm{PBM}(N_j)$
    $\mathcal{E}_\nu \leftarrow \|u - u_\mathrm{p}\|$
    **if** $\nu > 0$ **then**
        $\lambda_\nu \leftarrow \log_\zeta (\mathcal{E}_{\nu-1}/\mathcal{E}_\nu)$
    **end**
    $N_j \leftarrow \zeta \cdot N_j$
**end**
If the true $\lambda$ is known, assert $\lambda \approx \lambda_\nu \ \forall \ \nu \in \{1, \ldots, N_\nu\}$.
```

4.5.2. Procedure for Grid Refinement Studies with Data-Driven or Hybrid Models

Here, we will describe the procedure used to conduct the two grid refinement studies presented in the next section. In addition to considering PBM, as is customary, these studies also consider DDM and HAM models. Since DDM and HAM models require training, it might not be possible to use such models with different grids without retraining them. This raises some questions regarding how grid refinements should be performed when considering DDM or HAM models. Below, we will simply describe the procedure we have used in our two studies; a more in-depth discussion can be found in Appendix C.

For the first study, we use Solution P1 as the reference solution, while Solution P3 is used as the reference solution for the second study. In both studies, we assume that the full physics is known to our PBM and HAM models. (As always, the DDM has to learn the full physics from scratch.) We perform $N_\nu = 5$ refinements in each study, starting with $N_j = 5$ grid cells for the coarsest grid and increasing the number of grid cells by a factor $\zeta = 3$ for each refinement. The DDM and HAM models use the DNN architecture and hyperparameters described in Section 4.1.4 at all resolutions in both studies. However, note that the dimensionality of the DNNs' input and output layers must be different for different discretizations, since the number of nodes in these layers must match the number of components in the discrete temperature profiles. That is, the DNN architecture must be altered slightly from one refinement level to the next. Consequently, the DNNs must be retrained from scratch at every refinement level. We train the DNNs using the procedure described in Section 4.1.3, such that the DNNs are trained and validated using data corresponding to $\alpha \in \mathcal{A}_\mathrm{train}$ and $\alpha \in \mathcal{A}_\mathrm{val}$, respectively. The complete models are then evaluated using data corresponding to $\alpha \in \mathcal{A}_\mathrm{test}$. An overview of the procedure used to conduct our two grid refinement studies is presented in Algorithm 2.

Algorithm 2: Procedure used in the two grid refinements conducted as part of the present work.

```
Set the initial number of grid cells, $N_j = 5$.
Set the grid refinement factor, $\zeta = 3$.
Set the number of grid refinements to perform, $N_\nu = 5$.
for $\nu$ in $\{0, \dots, N_\nu\}$ do
    Train the DNNs of HAM and DDM for the current $N_j$ using data
      corresponding to $\alpha \in \mathcal{A}_{\text{train}}$.
    for $\alpha$ in $\mathcal{A}_{\text{test}}$ do
        Compute $\boldsymbol{T}_{\text{p}}^{N_t-1}$, $\boldsymbol{T}_{\text{d}}^{N_t-1}$ and $\boldsymbol{T}_{\text{h}}^{N_t-1}$ for the current $N_j$.
        Compute the relative $\ell_2$-errors of $\boldsymbol{T}_{\text{p}}^{N_t-1}$, $\boldsymbol{T}_{\text{d}}^{N_t-1}$, $\boldsymbol{T}_{\text{h}}^{N_t-1}$ w.r.t. $\boldsymbol{T}_{\text{ref}}^{N_t-1}$.
    end
    $N_j \leftarrow \zeta \cdot N_j$
end
```

4.5.3. Results and Discussion

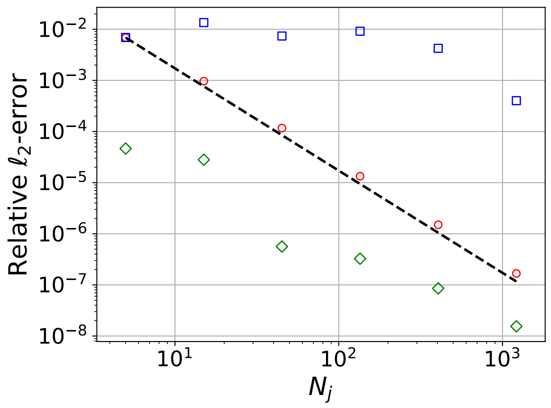
The results of the grid refinement study considering Solution P1 are shown in Figure 4.45, while Figure 4.46 present the results of the study considering Solution P3.

In Figure 4.45, we see that the error of the PBM is a steadily decreasing function of N_j. For all four α-values, it closely follows the dashed line indicating second order accuracy. This is the expected behaviour, since the Implicit Euler FVM is second order accurate. In contrast, there does not seem to be any clear trend in the DDM errors. They appear to be mostly constant, albeit with some noise. The noise is particularly prominent for $\alpha = 1.5$. A possible explanation for the noise is that the chosen DNN hyperparameters might incidentally be better suited for handling data at certain resolutions.

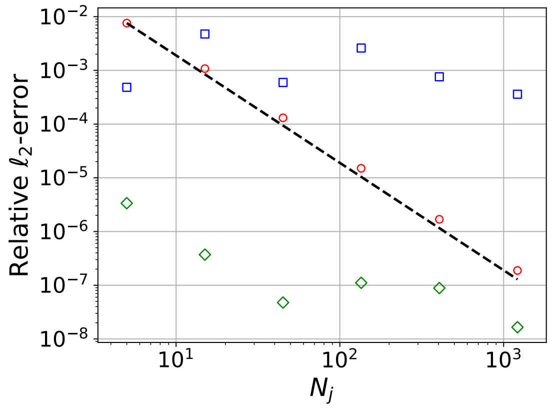
Though we have no results from the literature to compare with, it appears reasonable that the DDM errors are more or less independent of the spatial resolution. This is because the DDM always has to learn the full physics, regardless of the resolution of the data. Thus, the DDM's accuracy will only increase for higher resolutions if increasing the resolution uncovers new information in its training data. This could conceivably occur e.g. in turbulent flow scenarios, but it is not the case for the smooth temperature profiles considered here.

Looking at the HAM errors in Figure 4.45, we observe a behaviour that is somewhere in between PBM and DDM. This is perhaps not so surprising, considering that our HAM model is a hybrid of PBM and DDM. For all four $\alpha \in \mathcal{A}_{\text{test}}$, we observe that the ℓ_2-error of the HAM predictions is a decreasing function of N_j. However, it does not decrease at a steady rate like the PBM error. Just like for the DDM, this might be because the DNN is incidentally better configured for some resolutions than for others. We also observe that the HAM error for $\alpha = 1.5$ is not decreasing for large N_j. This could possibly be attributed to the finite numerical precision of the data type (float64) used to conduct the studies. A relative ℓ_2-error of roughly 1e-8 implies that the components of the difference vector $\boldsymbol{T}_{\text{h}}^{N_t-1} - \boldsymbol{T}_{\text{ref}}^{N_t-1}$ are of the order 1e-16, which is also the order of machine epsilon for float64. For this reason, we do not put much emphasis on the results for $\alpha = 1.5$ and large N_j.

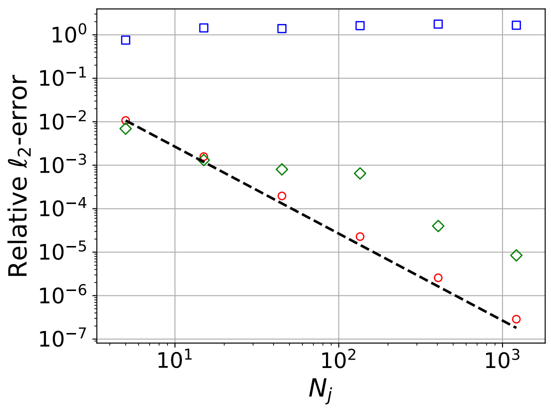
Figure 4.46 tells much the same story as Figure 4.45. The most prominent difference is perhaps that the PBM errors level off at large N_j. Such behaviour occurs when the temporal discretization error becomes non-negligible, such that Equation (4.10) does



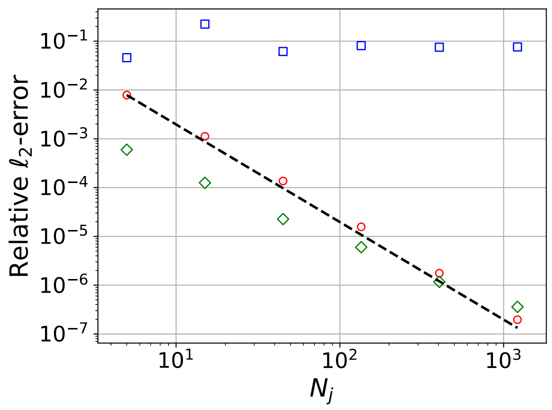
(a) $\alpha = 0.7$



(b) $\alpha = 1.5$



(c) $\alpha = -0.5$



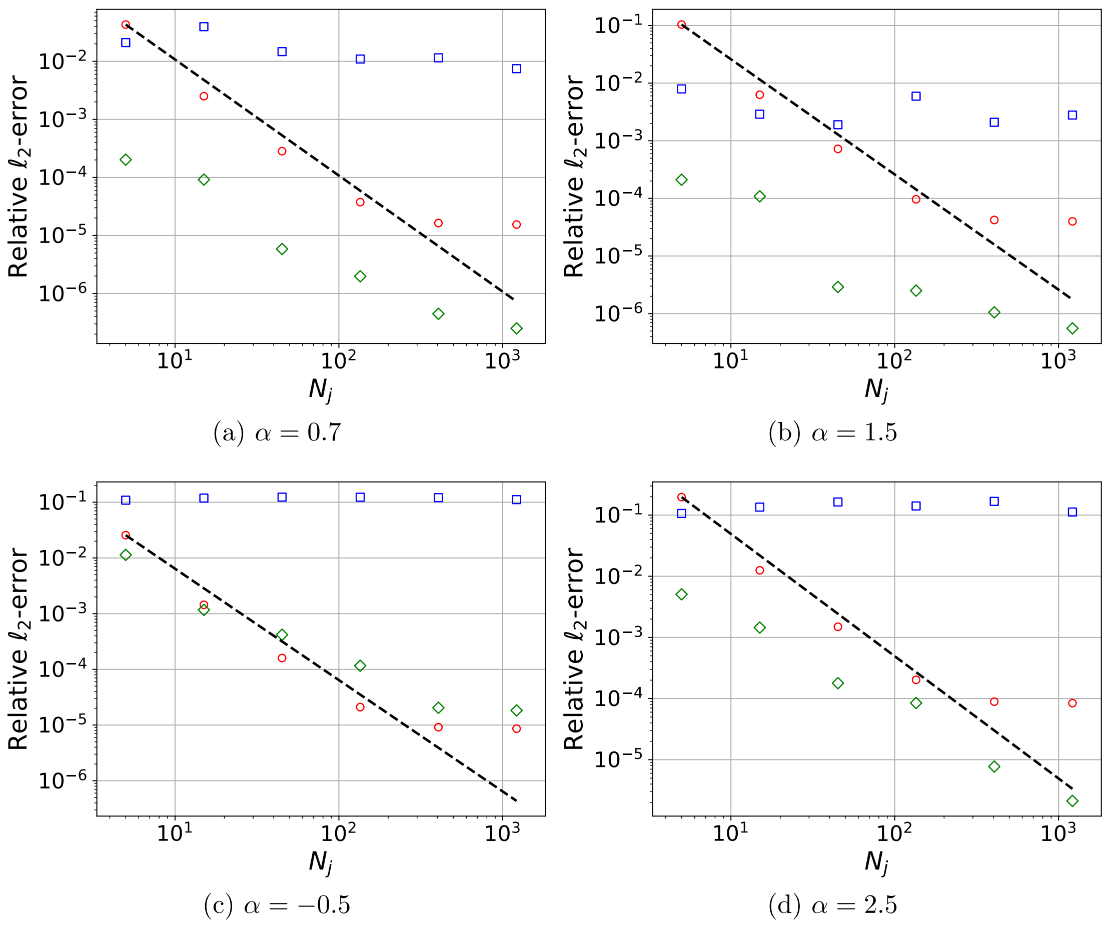
(d) $\alpha = 2.5$

Figure 4.45.: Solution P1, spatial grid refinement: Relative ℓ_2-error of predicted temperature at the final time level, shown as a function of the number of grid cells N_j for all $\alpha \in \mathcal{A}_{\text{test}}$ (○ PBM, □ DDM, ◇ HAM, - - 2nd order accuracy).

not hold. This observation is therefore not a reason for concern. As for the DDM, we observe that its ℓ_2-error is virtually independent of N_j, just like in the other study. Another observation that is common between the two studies is that HAM improves as N_j increases, and we attribute this to the improved accuracy of the PBM for increasing N_j. However, unlike the PBM, the HAM model does not improve at a steady rate.

In conclusion, the results indicate that grid refinement studies do not facilitate the establishment of en empirical order of accuracy for data-driven or hybrid models, because their errors do not decrease at an (approximately) constant rate. As such, grid refinement studies (at least in their present form[19]) are not as valuable for verification of DDM or HAM models as for PBMs. However, the studies presented here have illustrated an important characteristic of CoSTA-based HAM models: Improving the accuracy of a PBM also improves the accuracy of HAM models based on that PBM. This is a powerful result, because it implies that CoSTA-based models benefit from advances in physic-based modelling, e.g. as a result of novel experimental research. Interestingly, advances in physics-based modelling may also come from interpreting the HAM model itself, as we will discuss in the next section.

[19]We cannot exclude the possibility that some of the alternative approaches to grid refinement studies discussed in Appendix C would provide different results.



(a) $\alpha = 0.7$

(b) $\alpha = 1.5$

(c) $\alpha = -0.5$

(d) $\alpha = 2.5$

Figure 4.46.: Solution P3, spatial grid refinement: Relative ℓ_2-error of predicted temperature at the final time level, shown as a function of the number of grid cells N_j for all $\alpha \in \mathcal{A}_{\text{test}}$ (○ PBM, □ DDM, ◇ HAM, - - 2nd order accuracy).

4.6. Interpretability of the Corrective Source Term

A strong-point of CoSTA-based HAM is that the corrective source term can be interpreted in a physics-based framework. For heat transfer problems like the ones considered in this chapter, the most intuitive interpretation is that the corrective source term can be seen as a correction to the modelling of the physical source term stemming from the internal heat generation rate P. However, the corrective source term can also be interpreted as a correction to other physical parameters, such as the conductivity k or the density ρ. Below, we demonstrate how to recover an unknown heat generation rate P and an unknown conductivity profile k using data from experiments considered in Sections 4.2.2 and 4.2.3. We also discuss how an unknown density ρ or heat capacity c_V can be recovered in similar ways. These ways of interpreting the corrective source term have great implications for the trustworthiness of CoSTA-based HAM, and are thus highly relevant for the third research question listed in Section 1.2.

4.6.1. Recovering an Unknown Source Term

Suppose we approximate the true heat generation rate P in the heat equation (2.8) with $\widetilde{P} = P - \epsilon_P$, where $\epsilon_P \neq 0$ is the error of the approximation. Inserting $\widetilde{P} + \epsilon_P$ for P in

Equation (2.8), we obtain

$$\int\limits_V \rho c_V \frac{\partial T}{\partial t} \mathrm{d}V = \left(kA\frac{\partial T}{\partial x}\right)_e - \left(kA\frac{\partial T}{\partial x}\right)_w + \int\limits_V P \,\mathrm{d}V \tag{4.11}$$

$$= \left(kA\frac{\partial T}{\partial x}\right)_e - \left(kA\frac{\partial T}{\partial x}\right)_w + \int\limits_V \widetilde{P} \,\mathrm{d}V + \int\limits_V \epsilon_P \,\mathrm{d}V. \tag{4.12}$$

Following the discretization procedure used in Section 2.2.3, we can discretize the above equation as follows:

$$\mathbb{A}\boldsymbol{T}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}^n\right) + \Delta t \tilde{\boldsymbol{\sigma}}_P. \tag{4.13}$$

Here, $\mathbb{A}$, $\boldsymbol{T}$ and $\boldsymbol{b}$ are defined as in Equation (2.18), and $\tilde{\boldsymbol{\sigma}}_P = \boldsymbol{\epsilon}_P/(\rho c_V)$, where $\boldsymbol{\epsilon}_P = [\epsilon_P(x_1), \dots, \epsilon_P(x_{N_j})]$ and $x_1, \dots, x_{N_j}$ are the grid nodes used to define $\boldsymbol{T}$. Comparing Equation (4.13) to Equation (3.15), we can see that there is a clear connection between $\tilde{\boldsymbol{\sigma}}_P$ and the corrective source term $\hat{\boldsymbol{\sigma}}$ used in CoSTA. In fact, it appears that we can simply write $\Delta t \tilde{\boldsymbol{\sigma}}_P = \hat{\boldsymbol{\sigma}}$. However, this equality does not hold in general, as $\hat{\boldsymbol{\sigma}}$ is defined to correct for any discretization error in the PBM as well as possible modelling error, while $\tilde{\boldsymbol{\sigma}}_P$ only corrects errors in the modelling of P. To make progress, we split $\hat{\boldsymbol{\sigma}}$ into two contributions, $\hat{\boldsymbol{\sigma}}_{\text{discr}}$ and $\hat{\boldsymbol{\sigma}}_{\text{mod}}$, corresponding to discretization error and modelling error, respectively. If we now assume that $\hat{\boldsymbol{\sigma}}_{\text{discr}} \ll \hat{\boldsymbol{\sigma}}_{\text{mod}}$, then $\hat{\boldsymbol{\sigma}} \approx \hat{\boldsymbol{\sigma}}_{\text{mod}}$. Furthermore, if $\hat{\boldsymbol{\sigma}}_{\text{mod}}$ stems from an incorrect value of P, we then have

$$\hat{\boldsymbol{\sigma}} \approx \Delta t \tilde{\boldsymbol{\sigma}}_P \quad \Longleftrightarrow \quad \frac{1}{\Delta t}\hat{\boldsymbol{\sigma}} \approx \tilde{\boldsymbol{\sigma}}_P = \frac{\boldsymbol{\epsilon}_P}{\rho c_V}. \tag{4.14}$$

Using the relations above, we can define the DNN-predicted error in P, $\hat{\boldsymbol{\epsilon}}_P$, as

$$\hat{\boldsymbol{\epsilon}}_P = \frac{\rho c_V}{\Delta t}\hat{\boldsymbol{\sigma}}_{\text{NN}} \approx \boldsymbol{\epsilon}_P. \tag{4.15}$$

The take-away point here is that, given the DNN-predicted corrective source term $\hat{\boldsymbol{\sigma}}_{\text{NN}}$, we can compute an approximation $\hat{\boldsymbol{\epsilon}}_P$ of the true error $\boldsymbol{\epsilon}_P$ in the modelling of the heat generation rate P.

In Section 4.2.2, we considered a series of 1D experiments with significant modelling error due to incorrect modelling of P. We will now use data from the experiment considering Solution P1 therein to demonstrate how an unknown source term can be recovered from the DNN-generated corrective source term $\hat{\boldsymbol{\sigma}}_{\text{NN}}$.

The first step in recovering the unknown source term is to compute $\hat{\boldsymbol{\epsilon}}_P$ given $\hat{\boldsymbol{\sigma}}_{\text{NN}}$ using Equation (4.15). For this particular example, $\hat{\boldsymbol{\sigma}}_{\text{NN}}$ is the same for all time levels (bar some random noise), so it is not important which time level we take data from. However, for completeness, we state that we use data from the final time $t_{\text{end}} = 5.0\,\text{s}$. In Figure 4.47, we have plotted the DNN-predicted error $\hat{\boldsymbol{\epsilon}}_P$ computed using $\hat{\boldsymbol{\sigma}}_{\text{NN}}$ for $\alpha = 0.7$ and $\alpha = 1.5$.

In the second step, we analyze the data from Figure 4.47. The analysis will inherently depend on what we know about the system at hand and about the PBM used. As an example, we will here assume that the we know that some temporally and spatially uniform heating/cooling is applied to the system, and that this heating/cooling is unaccounted for by the PBM. Furthermore, we assume that we know that the heating/cooling is related to α in some unknown way. For both α-values, Figure 4.47 shows that $\hat{\boldsymbol{\epsilon}}_P$ is roughly constant throughout the entire domain except at the grid nodes closest to the boundary. In this example, we have no *a priori* knowledge indicating that

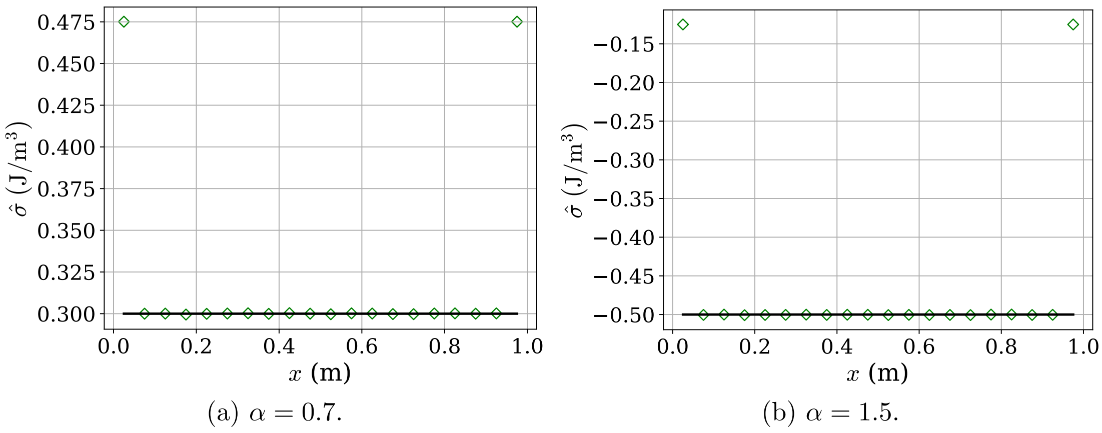


Figure 4.47.: Solution P1, interpolation: Corrective source terms predicted by the DNN of HAM at the final time level (◇), and the constant function $1 - \alpha$, which corresponds to the unknown source term of Solution P1 (—). The data for this figure was generated using the same HAM model as was used to generate Figure 4.7.

there is anything interesting going on at the boundaries, and we therefore disregard the boundary behaviour as being caused by numerical effects unrelated to the unmodelled heating/cooling.[20] Instead, we focus on the fact that $\hat{\boldsymbol{\epsilon}}_P$ is roughly constant in both time and space, which makes it likely that it can account for the heating/cooling not included in the PBM. From Figure 4.47, it appears that we have $\epsilon_P(\alpha = 0.7) \approx 0.3$ and $\epsilon_P(\alpha = 1.5) \approx -0.5$, which is consistent with $\epsilon_P = 1 - \alpha$. Since the PBM discussed in Section 4.2.2 has $\widetilde{P} = 0$, this implies $P = \widetilde{P} + \epsilon_P = 1 - \alpha$, which is indeed the true source term of Solution P1 (cf. Table 4.1). In an application context, where we would not know for sure that we had recovered the exact ϵ_P, we would still know whether the obtained ϵ_P is consistent with the postulated behaviour of the unknown P. This could serve as a sanity check for the DNN in the HAM model. However, at present, this sanity check requires human interaction at several levels, such as when we decided to ignore the behaviour of $\hat{\boldsymbol{\sigma}}_{\text{NN}}$ near the boundaries. Further research into automating this sanity check would be useful.

Of course, in a high-stakes application, we would require more than two data points before claiming $\epsilon_P \approx 1 - \alpha$ as we did above. For example, $\epsilon_P = 2.05 - 3.2\alpha + \alpha^2$ is also reconcilable with the Figure 4.47. Furthermore, if ϵ_P appears to be varying in space and/or time, we would have to use some regression technique to find a probable explicit expression for ϵ_P. However, the underlying principles remain the same as those used above also in more complex scenarios. Furthermore, we highlight that the procedure used above is also valid for higher-dimensional problems, as the error ϵ_P will be captured by an extra term on the right-hand side of the FVM's matrix form regardless of the problem's dimensionality.

Even without obtaining any general explicit expression for $\hat{\boldsymbol{\epsilon}}_P$ through regression, calculating $\hat{\boldsymbol{\epsilon}}_P$ at regular time intervals might still give important qualitative insight into the behaviour of a CoSTA-based HAM model's DNN. For instance if it is known that the system is being heated by some unmodelled heat source, then we know that $\widetilde{P}$ is

[20]Since this is a synthesized example, we know that this is a valid assumptions. In a real-world application scenario, further investigations may be warranted before such an assumption can be made with a sufficient degree of certainty.

too low, which corresponds to positive ϵ_P. In such a scenario, it would thus be grounds for concern if the DNN-predicted error $\hat{\boldsymbol{\epsilon}}_P$ has components that are significantly negative. Similarly, if we know that the heat added to a system is bounded, then we know that the DNN-corrected source term

$$\hat{\boldsymbol{P}} = \widetilde{\boldsymbol{P}} + \hat{\boldsymbol{\epsilon}}_P = \widetilde{\boldsymbol{P}} + \frac{\rho c_V}{\Delta t}\hat{\boldsymbol{\sigma}}_{\mathrm{NN}} \tag{4.16}$$

must also be bounded. At any given time level, we can check if the appropriate bounds are respected, and this can serve as an automatic sanity check for the DNN. Such sanity checks make the complete model far more trustworthy. Note however that care must be taken when defining the bounds. Some numerical methods do not conserve energy, and when a CoSTA-based HAM model utilizes such a solver in its PBM, it may be desirable for the corrective source term to add energy to the system. Thus, determining the appropriate bounds must done with care on a case-by-case basis. The important point here is that CoSTA facilitates application-specific sanity checks for its DNN when such checks are appropriate.

Aside from providing valuable sanity checks for the DNN, the approach outlined above can also be used to define a new and improved PBM. Instead of studying the DNN-predicted error in P, as defined in Equation (4.15), it can be fruitful to look deeper into the right-hand side of Equation (4.14), which defines an approximate relation between the *true* error in the modelling of P and the *true* corrective source term used to train the DNN. We now postulate that this approximate relation is instead an equality (which is true if incorrect modelling of P is our only source of error), such that we have

$$\boldsymbol{\epsilon}_P = \frac{\rho c_V}{\Delta t}\hat{\boldsymbol{\sigma}}. \tag{4.17}$$

We can then take all the true corrective source terms $\hat{\boldsymbol{\sigma}}$ from the DNN training data and use the equation above to calculate the corresponding $\boldsymbol{\epsilon}_P$. Subsequently, it is possible to apply some regression technique, e.g. symbolic regression, to the calculated errors $\boldsymbol{\epsilon}_P$. If successful, this would yield an explicit formula for ϵ_P. Using this formula, it is possible to define an alternative PBM where P is approximated by $\widetilde{P}+\epsilon_P$ rather than just $\widetilde{P}$. If this alternative PBM is found to outperform the original PBM, one may consider updating the PBM in the HAM accordingly. While this would require the DNN to be retrained, it would also make the resulting HAM model more powerful than the previous one, since improving the PBM also improves the full HAM model (cf. Section 4.5). Updating the PBM is especially relevant if the regressed ϵ_P has a meaningful physical interpretation, as that would increase the trustworthiness of the regressed correction significantly. In conclusion, the CoSTA framework not only facilitates the correction of a PBM using data-driven techniques – it also facilitates improving the original PBM, thereby also improving the full HAM model.

4.6.2. Recovering an Unknown Conductivity

We will now consider how an unknown conductivity can be recovered using the CoSTA framework. Suppose that the true conductivity k of a system is approximated by some $\tilde{k} = k - \epsilon_k$, where $\epsilon_k \neq 0$ is the error of the approximation (analogously to ϵ_P in the previous section). We consider here an approach that is different from the one used above in the sense that we use the differential form of the heat equation – not the integral form – as our starting point. The benefit of this approach is that it enables us to compute an explicit expression for ϵ_k for 1D problems. To begin, we insert $\tilde{k} + \epsilon_k$ into the one-dimensional heat equation on differential form to obtain

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + P \tag{4.18}$$

$$= \frac{\partial}{\partial x}\left(\left(\tilde{k} + \epsilon_k\right)\frac{\partial T}{\partial x}\right) + P \tag{4.19}$$

$$= \frac{\partial}{\partial x}\left(\tilde{k}\frac{\partial T}{\partial x}\right) + P + \frac{\partial}{\partial x}\left(\epsilon_k\frac{\partial T}{\partial x}\right) \tag{4.20}$$

$$\implies \quad \left(\frac{\partial}{\partial t} - \frac{\partial}{\partial x}\left(\tilde{k}\frac{\partial}{\partial x}\right)\right)T = P + \frac{\partial}{\partial x}\left(\epsilon_k\frac{\partial T}{\partial x}\right). \tag{4.21}$$

Comparing to Equation (3.5), we see that

$$\hat{\sigma} = \frac{\partial}{\partial x}\left(\epsilon_k\frac{\partial T}{\partial x}\right). \tag{4.22}$$

Under the assumption that $\partial T/\partial x \neq 0$, the equation above can be solved for ϵ_k, such that we obtain the following explicit formula:

$$\epsilon_k = \left(\frac{\partial T}{\partial x}\right)^{-1} \int_{x_a}^{x} \hat{\sigma}(x')\,\mathrm{d}x'. \tag{4.23}$$

Now, given a DNN-predicted corrective source term $\hat{\boldsymbol{\sigma}}^n_{\text{NN}}$ and the corresponding predicted temperature profile $\boldsymbol{T}^n_{\text{h}}$, we can compute the DNN-predicted error $\hat{\boldsymbol{\epsilon}}^n_k$ in the modelling of the conductivity k. At any time level n, the components of $\hat{\boldsymbol{\epsilon}}^n_k$ are given by

$$(\hat{\epsilon}^n_k)_j = \frac{(T^n_{\text{h}})_{j+1} - 2(T^n_{\text{h}})_j + (T^n_{\text{h}})_{j-1}}{(x_{j+1} - x_j)(x_j - x_{j-1})} \int_{x_a}^{x_j} \mathcal{I}[\hat{\boldsymbol{\sigma}}^n_{\text{NN}}](x')\,\mathrm{d}x', \qquad j = 1, \ldots, N_j. \tag{4.24}$$

Here, $\mathcal{I}[\hat{\boldsymbol{\sigma}}^n_{\text{NN}}]$ is some integrable spatial interpolant of $\hat{\boldsymbol{\sigma}}^n_{\text{NN}}$, and we have approximated $\partial T/\partial x$ with a second order central difference approximation using $\boldsymbol{T}^n_{\text{h}}$. Fifth order Gaussian interpolation is a scheme which we have found to work well for interpolating $\hat{\boldsymbol{\sigma}}^n_{\text{NN}}$. Before moving on to some practical examples, we highlight that Equation (4.24) implicitly assumes that the modelling error due to the incorrect conductivity is much larger than other errors affecting the model.

In Section 4.2.3, we considered a series of numerical experiments with unknown conductivity. We will now attempt to recover the conductivities corresponding to Solution k1 and Solution k2 in Table 4.1 using data from these experiments. More specifically, we will insert the corrective source term $\hat{\boldsymbol{\sigma}}^{N_t-1}_{\text{NN}}$ and the predicted temperature $\boldsymbol{T}^{N_t-1}_{\text{h}}$ into Equation (4.24) to predict the conductivity error $\hat{\boldsymbol{\epsilon}}_k$. The DNN-corrected conductivity at the final time $t_{\text{end}} = 5.0\,\text{s}$ is then calculated using the formula $\hat{\boldsymbol{k}} = \tilde{\boldsymbol{k}} + \hat{\boldsymbol{\epsilon}}_{\boldsymbol{k}}$. We illustrate $\hat{\boldsymbol{k}}$ for Solutions k1 and k2 in Figures 4.48 and 4.49, respectively. As can be seen from the figures, the corrected conductivities $\hat{\boldsymbol{k}}$ (green diamonds) correspond very well to the true conductivities (black line). It is especially noteworthy that $\hat{\boldsymbol{k}}$ is so similar to k for Solution k2, where k is proportional to T and therefore varies not only in space, but also in time.

Despite the good results illustrated in Figures 4.48 and 4.49, the approach we used above has several weakness which must be highlighted. First of all, Equation (4.24) is numerically unstable for small $\partial T/\partial x$. Secondly, due to the outer spacial derivative

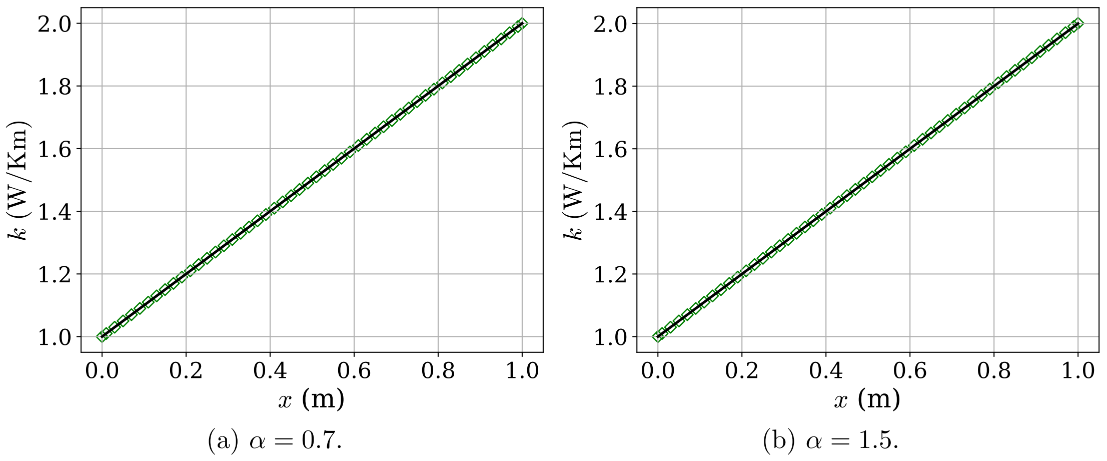


Figure 4.48.: Solution k1, interpolation: Conductivity profiles at the final time level recovered using the corrective source term predicted by HAM (◇), and true conductivity profiles of Solution k1 (—) for $\alpha \in \{0.7, 1.5\}$. The data for this figure was generated using the same HAM model as was used to generate Figure 4.15.

in the differential form of the heat equation, any constant terms in $k(\partial T/\partial x)$ will not have any influence on the true corrective source term $\hat{\sigma}$ and they will therefore not be reflected in the DNN-generated corrective source term $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}$ either. This means that, in some scenarios, it is impossible to recover the true conductivity exactly. For example, if T is linear in space, any unknown constant term in k cannot be recovered using Equation (4.24). Consequently, it is e.g. impossible to recover the ratio between the two constant conductivity levels used to define Solution k3 from $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}$. However, it would have been possible if we had chosen a different manufactured solution such that we did not have $k(\partial T/\partial x) = 0$ everywhere except at the discontinuity.

Another weakness of the approach used above is that it cannot be generally extended to higher-dimensional problems. As an example, we use $\tilde{k} = k - \epsilon_k$ in the 2D heat equation on differential form below.

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right) + P \tag{4.25}$$

$$= \frac{\partial}{\partial x}\left(\left(\tilde{k} + \epsilon_k\right)\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\left(\tilde{k} + \epsilon_k\right)\frac{\partial T}{\partial y}\right) + P \tag{4.26}$$

$$= \frac{\partial}{\partial x}\left(\tilde{k}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\tilde{k}\frac{\partial T}{\partial y}\right) + P + \frac{\partial}{\partial x}\left(\epsilon_k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\epsilon_k\frac{\partial T}{\partial y}\right) \tag{4.27}$$

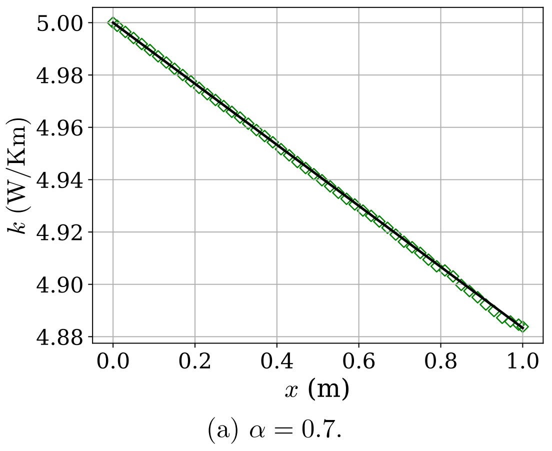
Comparing the above to Equation (3.5) and Equation (4.21), it is clear that we have

$$\hat{\sigma} = \frac{\partial}{\partial x}\left(\epsilon_k\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\epsilon_k\frac{\partial T}{\partial y}\right)$$

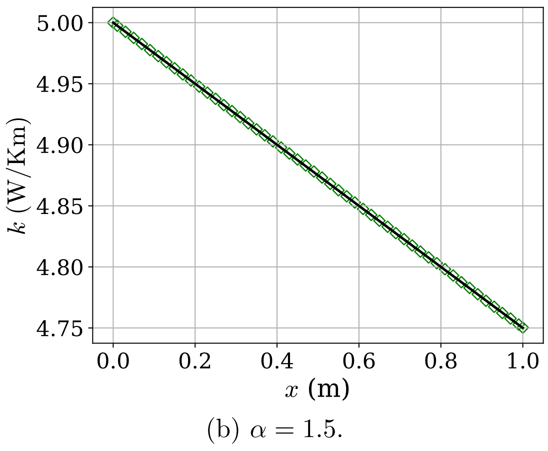
which can be rewritten as

$$\int_{y_c}^{y}\int_{x_a}^{x} \hat{\sigma}(x', y')\,\mathrm{d}x'\mathrm{d}y' = \int_{y_c}^{y} \epsilon_k\frac{\partial T}{\partial x}\mathrm{d}y' + \int_{x_a}^{x} \epsilon_k\frac{\partial T}{\partial y}\mathrm{d}x'. \tag{4.28}$$

Unfortunately, Equation (4.28) cannot be solved for ϵ_k in general.



(a) $\alpha = 0.7$.



(b) $\alpha = 1.5$.

Figure 4.49.: Solution k2, interpolation: Conductivity profiles at the final time level recovered using the corrective source term predicted by HAM (◇), and true conductivity profiles of Solution k2 (—) for $\alpha \in \{0.7, 1.5\}$. The data for this figure was generated using the same HAM model as was used to generate Figure 4.16.

One possible fix for higher-dimensional problems is to instead use the integral form of the heat equation as the starting point. For example, in 2D, using $k = \tilde{k} + \epsilon_k$ in the heat equation on integral form yields

$$\begin{aligned}
\int\limits_V \rho c_V \frac{\partial T}{\partial t} \mathrm{d}V &= \left(kA\frac{\partial T}{\partial x}\right)_e - \left(kA\frac{\partial T}{\partial x}\right)_w + \left(kA\frac{\partial T}{\partial y}\right)_n - \left(kA\frac{\partial T}{\partial y}\right)_s + \int\limits_V P\,\mathrm{d}V \\
&= \left((\tilde{k}+\epsilon_k)A\frac{\partial T}{\partial x}\right)_e - \left((\tilde{k}+\epsilon_k)A\frac{\partial T}{\partial x}\right)_w \\
&\quad + \left((\tilde{k}+\epsilon_k)A\frac{\partial T}{\partial y}\right)_n - \left((\tilde{k}+\epsilon_k)A\frac{\partial T}{\partial y}\right)_s + \int\limits_V P\,\mathrm{d}V \\
&= \left(\tilde{k}A\frac{\partial T}{\partial x}\right)_e - \left(\tilde{k}A\frac{\partial T}{\partial x}\right)_w + \left(\tilde{k}A\frac{\partial T}{\partial y}\right)_n - \left(\tilde{k}A\frac{\partial T}{\partial y}\right)_s + \int\limits_V P\,\mathrm{d}V \\
&\quad + \left(\epsilon_k A\frac{\partial T}{\partial x}\right)_e - \left(\epsilon_k A\frac{\partial T}{\partial x}\right)_w + \left(\epsilon_k A\frac{\partial T}{\partial y}\right)_n - \left(\epsilon_k A\frac{\partial T}{\partial y}\right)_s ,
\end{aligned}$$

from which it is clear that

$$\hat{\sigma} = \left(\epsilon_k A\frac{\partial T}{\partial x}\right)_e - \left(\epsilon_k A\frac{\partial T}{\partial x}\right)_w + \left(\epsilon_k A\frac{\partial T}{\partial y}\right)_n - \left(\epsilon_k A\frac{\partial T}{\partial y}\right)_s . \tag{4.29}$$

Now, to make use of the discrete DNN-predicted $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}$, we have to discretize the equation above. For the derivative of the temperature, we can use the same central difference approximations as we used for deriving the Implicit Euler FVM (cf. Equations (2.13) and (2.20)). For the conductivity modelling error ϵ_k, we suggest using arithmetic averaging because it is linear (which means we end up with a linear system of equations later). Using the index notation of Section 2.2.3, the averages read

$$(\epsilon_k)_{j+1/2,i} \approx \frac{(\epsilon_k)_{j+1,i} + (\epsilon_k)_{j,i}}{2}, \quad (\epsilon_k)_{j-1/2,i} \approx \frac{(\epsilon_k)_{j,i} + (\epsilon_k)_{j-1,i}}{2}, \tag{4.30}$$

$$(\epsilon_k)_{j,i+1/2} \approx \frac{(\epsilon_k)_{j,i+1} + (\epsilon_k)_{j,i}}{2}, \quad (\epsilon_k)_{j,i-1/2} \approx \frac{(\epsilon_k)_{j,i} + (\epsilon_k)_{j,i-1}}{2}. \tag{4.31}$$

Inserting these approximations into Equation (4.29) and making the replacements $\hat{\boldsymbol{\sigma}} \to \hat{\boldsymbol{\sigma}}_{\text{NN}}$, $\epsilon_k \to \hat{\epsilon_k}$ and $T \to T_{\text{h}}$, we obtain

$$\begin{aligned}(\hat{\sigma}_{\text{NN}})_{i,j} = {}& \frac{(\hat{\epsilon}_k)_{j+1,i} + (\hat{\epsilon}_k)_{j,i}}{2}\frac{(T_{\text{h}})_{j+1,i} - (T_{\text{h}})_{j,i}}{x_{j+1} - x_j} - \frac{(\hat{\epsilon}_k)_{j,i} + (\hat{\epsilon}_k)_{j-1,i}}{2}\frac{(T_{\text{h}})_{j,i} - (T_{\text{h}})_{j-1,i}}{x_j - x_{j-1}} \\ & \frac{(\hat{\epsilon}_k)_{j,i+1} + (\hat{\epsilon}_k)_{j,i}}{2}\frac{(T_{\text{h}})_{j,i+1} - (T_{\text{h}})_{j,i}}{y_{i+1} - y_i} - \frac{(\hat{\epsilon}_k)_{j,i} + (\hat{\epsilon}_k)_{j,i-1}}{2}\frac{(T_{\text{h}})_{j,i} - (T_{\text{h}})_{j,i-1}}{y_i - y_{i-1}},\end{aligned}$$

which constitutes a linear system of $N_j \cdot N_i$ equations. Unfortunately, it has $(N_j + 2) \cdot (N_i + 2)$ unknowns, and is thereby underdetermined. However, applying e.g. extrapolating boundary conditions[21] would make the system determined, thereby allowing us to compute $\hat{\boldsymbol{\epsilon}}_k$ at any time level given $\hat{\boldsymbol{\sigma}}_{\text{NN}}$ and $\boldsymbol{T}_{\text{h}}$ from the same time level. We recommend looking further into this approach to computing $\hat{\boldsymbol{\epsilon}}_k$ in future work regarding the interpretability of CoSTA-based HAM models.

4.6.3. Recovering an Unknown Density or Heat Capacity

In the present work, we have not conducted any experiments with an unknown density ρ or an unknown heat capacity c_V. Still, we provide the theoretical foundation for recovering an unknown density or heat capacity from the DNN-predicted corrective source term $\hat{\boldsymbol{\sigma}}_{\text{NN}}$. Below, we focus on recovering an unknown density, but the process for recovering an unknown heat capacity is entirely analogous. For simplicity, we also only cover 1D problems explicitly, but the procedure for higher-dimensional problems is also completely analogous.

Suppose that the true density ρ is approximated by some $\tilde{\rho} = \rho - \epsilon_\rho$, where $\epsilon_\rho \neq 0$ is the error of the approximation. Inserting this approximation into the 1D heat equation on integral form yields

$$\int\limits_V \rho c_V \frac{\partial T}{\partial t} \mathrm{d}V = \frac{\partial}{\partial x}\left(kA\frac{\partial T}{\partial x}\right)_e - \left(kA\frac{\partial T}{\partial x}\right)_w + \int\limits_V P \mathrm{d}V \tag{4.32}$$

$$\int\limits_V (\tilde{\rho} + \epsilon_\rho) c_V \frac{\partial T}{\partial t} \mathrm{d}V = \frac{\partial}{\partial x}\left(kA\frac{\partial T}{\partial x}\right)_e - \left(kA\frac{\partial T}{\partial x}\right)_w + \int\limits_V P \mathrm{d}V \tag{4.33}$$

$$\implies \int\limits_V \tilde{\rho} c_V \frac{\partial T}{\partial t} \mathrm{d}V = \frac{\partial}{\partial x}\left(kA\frac{\partial T}{\partial x}\right)_e - \left(kA\frac{\partial T}{\partial x}\right)_w + \int\limits_V P \mathrm{d}V - \int\limits_V \epsilon_\rho c_V \frac{\partial T}{\partial t} \mathrm{d}V \tag{4.34}$$

such that the continuous formulation of the corrective source term is

$$\hat{\sigma} = -\int\limits_V \epsilon_\rho c_V \frac{\partial T}{\partial t} \mathrm{d}V. \tag{4.35}$$

To express the predicted error in the modelling of ρ using the discrete quantities $\hat{\boldsymbol{\sigma}}_{\text{NN}}$ and $\boldsymbol{T}_{\text{h}}$, we need to discretize the equation above. This includes discretizing both the heat capacity and the error of the density. Here, we pick one mathematically convenient option for discretizing these quantities. First, we impose the following assumption on the cell-average of the right-hand-side integrand:

$$\frac{1}{\Delta x_j} \int\limits_{x_{j-1/2}}^{x_{j+1/2}} \epsilon_\rho c_V \frac{\partial T}{\partial t} \mathrm{d}x \approx \left(\epsilon_\rho c_V \frac{\partial T}{\partial t}\right)_j, \qquad j = 1, \ldots, N_j. \tag{4.36}$$

[21]Meaning that the value at any boundary node is set to the same value as that at its closest neighbouring node in the domain interior.

Then

$$\left(\frac{\partial T}{\partial t}\right)_j = -\frac{\hat{\sigma}_j}{(c_V)_j(\epsilon_\rho)_j} \tag{4.37}$$

To discretize the temporal derivative, we use the same discretization as for the Implicit Euler FVM (cf. Section 2.2.3), such that we obtain

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = -\frac{\hat{\sigma}_j^{n+1}}{(c_V)_j^{n+1}(\epsilon_\rho)_j^{n+1}}. \tag{4.38}$$

Making the substitutions $\hat{\sigma} \to \hat{\sigma}_{\text{NN}}$, $T \to T_\text{h}$ and $\epsilon_\rho \to \hat{\epsilon}_\rho$ yields

$$\frac{(T_\text{h})_j^{n+1} - (T_\text{h})_j^n}{\Delta t} = -\frac{(\hat{\sigma}_{\text{NN}})_j^{n+1}}{(c_V)_j^{n+1}(\hat{\epsilon}_\rho)_j^{n+1}}, \tag{4.39}$$

which can be rewritten as

$$(\hat{\epsilon}_\rho)_j^{n+1} = -\frac{\hat{\sigma}_j^{n+1}}{(c_V)_j^{n+1}}\frac{\Delta t}{(T_\text{h})_j^{n+1} - (T_\text{h})_j^n}. \tag{4.40}$$

The DNN-corrected density is then

$$\hat{\boldsymbol{\rho}} = \tilde{\boldsymbol{\rho}} + \hat{\boldsymbol{\epsilon}}_\rho, \tag{4.41}$$

where the components of $\hat{\boldsymbol{\epsilon}}_\rho$ are defined by Equation (4.40) for all $j = 1, \dots, N_j$. This DNN-corrected density can be used for sanity-checking the DNN of the CoSTA-based HAM model, or for updating the HAM model's internal PBM, analogously to the discussion on $\hat{\boldsymbol{\epsilon}}_P$ earlier in this section.

4.6.4. Concluding Remarks on CoSTA Interpretability

In this section, we have discussed how the DNN-generated corrective source term $\hat{\boldsymbol{\sigma}}_{\text{NN}}$ of CoSTA-based HAM models can be interpreted in a physical context. More specifically, we have used $\hat{\boldsymbol{\sigma}}_{\text{NN}}$ to recover unknown source terms and unknown conductivities using experimental data from earlier experiments, and we have discussed how analogous procedures can be used to recover other unknown parameters. Similar procedures can also be used for other physical systems than those governed by the heat equation. For example, in Appendix D, we explore how to recover an unknown wave speed for the wave equation.

The interpretability of the corrective source term has two main benefits. First of all, it facilitates sanity checks for the DNN used in the CoSTA-based HAM model. This results in increased trustworthiness, as divergent or otherwise unphysical behaviour can be detected automatically. Such automatic failure detection is especially important in high-stakes applications. Secondly, we have discussed how analyzing the corrective source term facilitates improvements in the PBM. As discussed in Section 4.5, improving the accuracy of the PBM generally leads to improved accuracy of the full CoSTA-based HAM model as well.

It should be mentioned that our approach to recovering unknown parameters assumes that the error of the unmodified PBM is dominated by a single error source. If several error sources contribute significantly to the total error, providing a useful analysis is more difficult. The reason for this is that one cannot determine the contribution of each error source to the corrective source term as a whole based on the corrective source term itself.

This is a well-known issue, which also occurs in traditional physics-based modelling. One topical example is climate modelling, where it can be difficult to accurately determine the specific influence of factors like solar irradiance variability, ENSO cycles[22] and volcanic activity on the atmospheric temperature at large (see e.g. (Lean, 2005; Lean and Rind, 2008) for interesting discussions on this topic). As such, this issue is not something that arises as a consequence of the CoSTA framework. Instead, it is a general issue encountered whenever one can only measure the cumulative effect of several partially understood phenomena. One should also keep in mind that useful analyses are still very much possible. The analyses are just more difficult to carry out, and may be influenced by larger uncertainty.

4.7. Review of Numerical Experiments

In this chapter, we have presented a variety of numerical experiments concerning unsteady heat transfer problems. In these experiments, traditional PBM and DDM are compared with the novel CoSTA-based HAM. To ensure a fair comparison, the physics-based and data-driven components of the HAM model are as similar to the stand-alone PBM and DDM as possible (cf. Section 4.1). Any performance difference between the HAM model and the other models can thus be attributed to the CoSTA framework, rather than the individual components of the HAM model. In Section 4.2, we considered 1D experiments, while 2D experiments were considered in Section 4.3. Three categories of *a priori* physics knowledge were studied in these experiments. In Section 4.2.1, we considered experiments where all physics was known, such that the goal of CoSTA was to reduce the discretization error of the PBM. Sections 4.2.2, 4.3.1, 4.2.3 and 4.3.2 were all devoted to experiments where some physics was unknown, such that CoSTA would additionally have to correct modelling error. In the former two sections, the modelling error resulted from an unknown source term, while an unknown conductivity was the source of error in the latter two sections. All the aforementioned experiments were aimed at answering our research questions regarding the accuracy and generalizability of CoSTA (cf. Section 1.2). The grid refinement studies conducted in Section 4.5 also contribute to answering these questions. The remaining two sections, Section 4.4 and Section 4.6, were aimed at answering the final research question concerning CoSTA's trustworthiness. In Section 4.4, we studied the inherent uncertainty of DDM and HAM models, while we discussed the interpretability of CoSTA's corrective source term in Section 4.6.

Below, we discuss and answer our three research questions (cf. Section 1.2) in light of the results and discussions presented in this chapter. We will thereafter highlight some important limitations of the present work, and briefly discuss how the present work could have been improved with the benefit of hindsight.

4.7.1. Answering the Research Questions

How does the predictive accuracy of a hybrid model using CoSTA compare to the accuracy of stand-alone PBM and DDM?

In the experiments of Sections 4.2 and 4.3, we have investigated the accuracy of PBM, DDM and CoSTA-based HAM for a variety of 1D and 2D heat transfer problems. A total of 28 interpolation scenarios and equally many extrapolation scenarios are considered in these experiments. The CoSTA-based HAM model is found to be the most accurate model in all 28 interpolation scenarios. In numerous interpolation scenarios,

[22]ENSO is short for El Niño Southern oscillation.

HAM outperforms the other models by several orders of magnitude in terms of relative ℓ_2-error. The HAM model is also significantly more accurate than the other models in 22 out of 28 extrapolation scenarios. HAM and PBM are jointly most accurate in 4 of the remaining 6 scenarios, while PBM is the most accurate model in the remaining two scenarios. Overall, it is clear that HAM is the most accurate model in the experiments described in Sections 4.2 and 4.3.

The grid refinement studies presented in Section 4.5 also allow us to compare the accuracy of the three models. Again, HAM is the most accurate in all interpolation scenarios. In the extrapolation scenarios, HAM significantly outperforms DDM, and is roughly as accurate as PBM. In general, it appears that PBM has a slight edge for scenarios with $\alpha = -0.5$ and large N_j. This is not surprising, given that, for both solutions considered in Section 4.5, $\alpha = -0.5$ yields temperature profiles that are qualitatively different to those seen during training. Additionally, the DNN hyperparameters were tuned on the basis of preliminary experiments with small N_j, so the HAM model is likely poorly tuned for use with large N_j. Still, HAM is the most accurate model in 15 out of the 24 extrapolation scenarios considered in Section 4.5.

In addition to accuracy in terms of ℓ_2-error, it is also interesting to compare how often the different models predict temperature profiles which are qualitatively incorrect. Of the 56 scenarios considered in Sections 4.2 and 4.3, DDM provides qualitatively incorrect predictions in 10 scenarios (mainly extrapolation scenarios), while PBM provides qualitatively incorrect predictions in 21 scenarios (all of which are scenarios with unknown physics). In comparison, HAM provides a qualitatively incorrect prediction in only one scenario. (In that scenario, the PBM and DDM predictions are also qualitatively incorrect, cf. Figure 4.13c.) Interestingly, CoSTA-based HAM is able to provide qualitatively correct predictions even when both PBM and DDM fail to do so (cf. Figure 4.11c).

In summary, CoSTA-based HAM is the most accurate model considered in our experiments. It provides better predictive accuracy than PBM and DDM, both in terms of relative ℓ_2-error and qualitative correctness. We have seen that this holds for both one- and two-dimensional problems. Additionally we have seen that CoSTA-based HAM is often able to improve the accuracy of PBM even when all physics is known *a priori*. This stands in contrast to DDM, which, in our experiments, only outperforms PBM in scenarios where some physics is unknown. Thus, for the purpose of predictive accuracy, we find that CoSTA better leverages the power of data-driven techniques than does stand-alone DDM.

How does the generalizability of a hybrid model using CoSTA compare to the generalizability of stand-alone PBM and DDM?

In Sections 4.2, 4.3 and 4.5, we have investigated the performance of PBM, DDM and CoSTA-based HAM in both interpolation and extrapolation scenarios. In analyzing the models' generalizability, we put emphasis on the results for the extrapolation scenarios. However, this analysis can be conducted from two different view-points. In Section 1.1, we defined a model's generalizability as its ability to solve a variety of problems without problem-specific fine-tuning. The two view-points differ in how they reconcile this definition with our experimental data. The first option is to look at how well the models retain their accuracy from the interpolation scenario in the extrapolation scenarios. The second option is to look at how well the models perform in the extrapolation scenarios without regarding their performance in the interpolation scenarios.

From the first point of view, PBM is the model with the highest generalizability. In all experiments we have conducted, the underlying assumptions of the PBM have been satisfied to the same extent for all scenarios within any one experiment. For example, in the experiments with unknown conductivity, the unknown conductivity was not more important for the overall behaviour in the system for some α-values than for others. Therefore, the accuracy of PBM is roughly the same in the extrapolation scenarios as in the interpolation scenarios. This level of performance consistency cannot be expected from DDM and HAM, as their DNNs are inherently affected by the bias-variance trade-off. Indeed, we observe that their ℓ_2-errors are consistently lower in the interpolation scenarios than in the extrapolation scenarios. However, the performance reduction is greater for DDM than for HAM. This is most clearly illustrated by the fact that DDM provides qualitatively incorrect predictions in a much higher number of extrapolation scenarios than does HAM.

From the second point of view, HAM exhibits the greatest generalizability of the three models. As mentioned in our discussion on model accuracy earlier, HAM is significantly more accurate than the other models in 22 out of 28 extrapolation scenarios, while being roughly as accurate as PBM in an additional 4 extrapolation scenarios. Furthermore, HAM also provides qualitatively correct predictions more frequently than the other two models. Thus, HAM is clearly the model that is most able to provide accurate predictions in the extrapolation scenarios. From this second point of view, one would therefore conclude that HAM exhibits the greatest generalizability.

It is an open question whether the first or second point of view is the "correct" one. Generally, the application will dictate which is the most reasonable. If consistent performance is more important than accuracy, then the first point of view is the most appropriate. However, if what matters most is the number of scenarios for which the model is able to provide accurate predictions, then the second point of view is a more natural choice. What is clear is that regardless of which point of view is chosen, CoSTA-based HAM generalizes better than pure DDM.

At last, we highlight the possibility of applying symbolic regression techniques to the corrective source term, as discussed briefly in Section 4.6. Cranmer et al. (2020) have shown that such an approach can be used to learn the complete governing equation. We therefore have reason to believe that symbolic regression can be used to learn an explicit representation of (some of) the physics captured by the corrective source term. Even if the symbolic regression does not capture all the physics of the corrective source term, obtaining explicit expressions for partial physics is also beneficial. The use of symbolic regression may increase generalizability of CoSTA-based models, as Cranmer et al. (2020) note that their regressed expressions generalize better than the DNNs to which the symbolic regression was applied. Combining CoSTA with symbolic regression is therefore an interesting research area which can be explored in the future.

Are predictions made by a CoSTA-based HAM model trustworthy?

In Section 1.1, we defined a model's trustworthiness as the extent to which the model can be explained and analyzed. Recall also that a CoSTA-based HAM model is created by augmenting a PBM with a corrective source term. We take it for granted that the original, non-augmented PBM can be derived from sound first principles, just like we derived the Implicit Euler FVM for the heat equation in Section 2.2. This derivation is in and of itself an explanation of the PBM. It also provides a framework for analyzing the PBM's behaviour, as different parameters with clear physical interpretations can be adjusted to analyze their impact on the model as a whole. For CoSTA, what remains is then to explain and analyze the corrective source term.

In Section 4.6, we discussed how the corrective source term of CoSTA-based HAM models can be interpreted in a physics context. We demonstrated how the corrective source term can be analyzed to recover unknown heating/cooling or unknown conductivity, and we also discussed how other unknown parameters could be recovered in the context of heat transfer problems. The presented recovery approach is completely general, and can also be applied to CoSTA-based HAM models for other systems. For example, Appendix D offers a discussion on how to recover an unknown wave speed from systems governed by the wave equation. As discussed in Section 4.6, the presented way of interpreting the corrective source term facilitates automatic performance monitoring, which is essential in safety-critical applications. It also facilitates updating the PBM of the CoSTA-based HAM model through e.g. symbolic regression. In addition to providing an accuracy boost to the HAM model (cf. the last paragraph of Section 4.5), this would also increase the model's interpretability and predictability, since we then get an explicit expression describing (some of) the unknown physics of the problem at hand.

Another important aspect of a model's trustworthiness is its susceptibility to noise. Random noise inherently degrades the explainability of a model, and thereby also its trustworthiness, because it introduces uncertainty in the model's predictions. Therefore, in an effort to develop trustworthy models, we want to keep the models' inherent uncertainty to a minimum. In Section 4.4, we presented results which are clearly indicative that the HAM models studied in our experiments have less inherent uncertainty than the DDM models. This conclusion is supported by observations from Section 4.3, where we noted that the DDM predictions are generally more noisy than the corresponding HAM predictions. Generally, our results suggest that CoSTA-based HAM yields a higher signal-to-noise ratio than DDM. This increases the interpretability of HAM in comparison to DDM, as whichever method is used for model analysis must deal with less noise. However, it is important to keep in mind that CoSTA-based HAM *is*, to some extent, influenced by noise. At present, we do not know how the long-term behaviour of CoSTA-based models is affected by the noise, and this lack of knowledge reduces the models' trustworthiness. More research into the long-term stability of CoSTA-based HAM models is therefore needed to unleash CoSTA's full potential in terms of trustworthiness. However, we observe that CoSTA is already more trustworthy than DDM, due to the reduced level of noise, the facilitated sanity-checks and the interpretability of the corrective source term.

At last, we highlight that the interpretability facilitated by the CoSTA-framework is complementary to ongoing research into DNN interpretability. In the computer science community, much research is devoted to finding systematic ways of interpreting the latent representations learnt by DNNs. That is to say, this work focuses on interpreting the inner workings of DNNs in an effort to understand why they make the predictions they do. Such knowledge is undoubtedly useful for any model based on DNNs, including CoSTA-based HAM models. In this work, we have explored a different approach to DNN interpretability, as we focus on the DNN's output rather than its internal latent representations. Such an approach has limited value for pure DDM, as it is difficult to analyze a time series of temperature profiles in any meaningful way without any further knowledge of the underlying physics. However, as we have demonstrated, such an approach is fruitful for CoSTA-based HAM models since the DNN output (the corrective source term) is then included in a physics-based framework. In this sense, CoSTA offers an extra layer of interpretability to data-driven techniques. It also offers an extra level of control over the impact of the data-driven techniques on the predictions made by the full model, because the corrective source term can be filtered, attenuated or otherwise modified before being coupled with the rest of the model. We envision that the improved interpretability and control offered by CoSTA can make data-driven techniques more attractive within high-stakes applications.

4.7.2. Limitations of the Present Work

From the results summarized above, it is clear that CoSTA has demonstrated many favourable characteristics in our numerical experiments. However, it is important to keep in mind that these experiments do not cover all relevant aspects of CoSTA for all conceivable applications. The major limitations of the present work are discussed below.

The most prominent limitation of the present work is that we have focused almost exclusively on heat transfer problems. This is not a necessary limitation, since CoSTA is a completely general framework, as discussed in Section 3.1. Therefore, it would be interesting to investigate to what extent the findings of the present chapter generalize to other kinds of problems. An initial effort in this direction is described in Appendix A, where we utilize CoSTA to reduce numerical diffusion in numerical solutions of the Euler equations used in e.g. gas dynamics research. However, Appendix A considers only a single experiment, and therefore has limited impact other than being a practical demonstration that CoSTA *can* be applied to other kinds of problems. More comprehensive studies are required to provide the full picture of CoSTA's performance for other applications.

Another important limitation of the experiments in the present chapter is that we have only considered a single PBM and a single DDM. Furthermore, the PBM and DDM considered here are both among the most basic models available. This was a deliberate choice by the author to make the demonstrations of CoSTA as simple as possible. Given the good results we have obtained for the basic models considered here, a natural next step is to investigate the performance of CoSTA using state-of-the-art data-driven techniques and physics-based models that are used in real-world industrial applications.

We highlight that the DDM and HAM models studied in this chapter both rely on the assumption that there is a strong correlation between data sampled at subsequent time levels. In cases where this assumption does not hold, there will not be a strong correlation between the target DNN input and output for these models. Their DNNs will then not have a well-defined learning problem, and will consequently learn poorly. The full models will then also perform poorly as a whole. An example of a problem where data from subsequent time levels is weakly correlated is the problem considered in Appendix A. Thus, while the CoSTA-framework itself is completely general, our particular implementation of the framework is not general. The particular issue noted here can be addressed by adopting a DNN architecture designed to handle temporally correlated data, such as recurrent neural networks.

Another example of our implementation not being general is that we have not informed the DNNs explicitly of α. This means that if different α-values yield the same initial temperature profile, then the DDM and HAM models considered here will not be able to differentiate between the two scenarios. For this reason, we could e.g. not use $T = x + \alpha t$ as a manufactured solution in our experiments, because we would then have $T(t = 0) = x$ for all α. We have taken care to choose manufactured solutions such that this has not been a problem in our experiments, but this is certainly something one would want to avoid in a general-purpose implementation. One option is to adopt a DNN-architecture like the one described by Pawar et al. (2021b) in their work on physics-guided machine learning. With such an architecture, the DNN can be explicitly informed of any important parameters in an efficient manner.

4.7.3. Lessons Learnt

With hindsight, one always discovers ways in which any research endeavour could have been improved. Below, we enumerate some additions and improvements the author would have made if the present work was to be repeated.

1. As discussed in Section 4.4, our method for measuring model uncertainty is somewhat unsatisfactory. The results of that section would have been more useful if we had e.g. used the ensembling approach advocated by Lakshminarayanan et al. (2017).
2. In Section 4.6, we considered *a posteriori* analysis of the corrective source term for the purpose of interpretability. However, other applications of *a posteriori* analysis have largely gone unexplored. Further considerations regarding the usefulness of such analysis would have been interesting.
3. Also in Section 4.6, we outlined an alternative approach to recovering an unknown conductivity profile. It would be valuable to know what quality of empirical results could be obtained using that method.
4. It would have increased the impact of this thesis if we had included at least one experiment where data-driven techniques specifically designed for time series forecasting had been used in the DDM and HAM models. For example, we could have used simple recurrent neural networks or temporal convolutional networks. This would have served two purposes. First of all, it would have been a clear demonstration of the fact that CoSTA is compatible with such techniques. Secondly, we could then have studied how improvements in DDM affect the performance of CoSTA-based HAM models, much like we investigated the effects of improving the PBM in Section 4.5.
5. The impact of the present work would also have increased if we would have been able to include some experiments based on real-world experimental data. A relevant experiment on heat transfer was originally intended to be discussed as part of this thesis, but the experimental rig was unfortunately not up and running in time for that to be possible.

5. Conclusion and Further Work

5.1. Conclusion

Predictive modelling techniques are paramount to our understanding and interaction with the world around us, as they play an important role within a wide array of industrial and scientific applications. Notable examples include climate modelling, metal production, off-shore wind and digital twins. Historically, most modelling techniques could be classified within one of two categories – physics-based modelling (PBM) and data-driven modelling (DDM) – with each category having its advantages and drawbacks. Due to its sound, first-principles foundation, PBM performs well in terms of generalizability and trustworthiness. However, these models are generally expensive in terms of computational resources, and they are static, meaning that they do not learn automatically from observations. In contrast, DDMs are inherently capable of learning from observations, and they exhibit high computational efficiency (at least for inference). However, they simultaneously suffer from limited generalizability due to the bias-variance trade-off. In addition, DDMs lack trustworthiness because their black-box-like nature inhibits rigorous analysis of the physics instilled into them. In response to the respective short-comings of pure PBM and DDM, a new modelling paradigm called *hybrid analysis and modelling* (HAM) has recently emerged. Within the HAM paradigm, PBM and DDM techniques are combined with the goal of creating hybrid models that retain the strengths of pure PBM and DDM, while eliminating their weaknesses. As such, successful development of modelling techniques within the HAM paradigm has the potential to positively impact numerous important applications, including those listed above.

In this thesis, we have formally introduced, justified and demonstrated the Corrective Source Term Approach (CoSTA) to HAM. The key concept of CoSTA is to augment the governing equation of a PBM with a corrective source term that is generated using DDM techniques. The purpose of the corrective source term is to account for any physics left unresolved by the original PBM. A similar concept has been utilized by Maulik et al. (2019) and Sirignano et al. (2020), who explore the use of deep neural networks (DNNs) to account for sub-grid processes in turbulence modelling. CoSTA can be seen as a generalization of these works, and is applicable to *any* deterministic system. Furthermore, as CoSTA does not require any *a priori* assumptions regarding the form of the error in the PBM, the corrective source term can account for *all* kinds of errors.

To investigate the accuracy and generalizability of CoSTA, we have conducted a series of numerical experiments on unsteady heat transfer problems. The experiments include both one- and two-dimensional problems, and consider varying levels of *a priori* knowledge of the physics to be modelled. Overall, CoSTA is found to be significantly more accurate than comparable PBM and DDM models – often outperforming the other models by several orders of magnitude in terms of relative ℓ_2-error. Furthermore, CoSTA provided qualitatively correct predictions in all but one of the scenarios considered in our experiments. This is again far better than DDM and PBM, which provided roughly 10 and roughly 20 qualitatively incorrect predictions, respectively. Interestingly, CoSTA has been found to provide accurate and qualitatively correct predictions even in scenarios where PBM and DDM both provide qualitatively incorrect predictions. In our experiments, CoSTA also demonstrates excellent generalizability in both interpolation and extrapolation scenarios. We draw particular attention to the observation that CoSTA generalizes significantly better than pure DDM. Overall, our experiments show that using PBM to account for known physics while allowing DDM to focus on learning what is truly unknown, is more efficient than using DDM to model *all* the physics of a system.

The findings listed above indicate that CoSTA can facilitate significant advances within a number of important applications. For example, the observation that CoSTA successfully reduces discretization error can be utilized to accelerate predictive models, as it enables the use of coarser discretizations for any desired level of accuracy. This is particularly useful for fluid flow simulations where state-of-the-art PBMs often take days to complete, even on high-end super-computers. Among the many applications that will benefit from speeding up fluid flow simulations, we highlight flow assurance and offshore wind as topical examples. Additionally, digital twin applications will also benefit greatly, as the need for real-time predictions necessitates the use of coarse discretizations with significant discretization error. The observation that CoSTA can reduce *modelling* error is also of great practical importance, e.g. for the production of aluminium (and other metals). In such applications, the (possibly non-linear) effects of temperature on material properties like conductivity and heat capacity, can be difficult to model using pure PBM. Other factors may also be difficult to capture using PBM, such as the effects of impurities within the metal and heat transfer into the production environment. CoSTA offers a way to account for these complex effects using data-driven techniques while maintaining the knowledge contained in the presently used PBM.

The applications used as examples above are all so-called *high-stakes* applications, where poor predictive modelling can result in unacceptable security risks and financial losses. Model trustworthiness is therefore of utmost importance in these applications, and they have consequently been dominated by pure PBM. However, we believe that CoSTA has the potential to change this, as it offers a new level of model explainability presently not found in DDM. Since CoSTA uses DDM techniques within a physics-based framework, the performance of the DDM can be analyzed within this physics-based framework. For example, we have demonstrated how the corrective source term can be used to recover unknown model parameters, such as an unknown heating/cooling term or an unknown conductivity profile. We have also discussed the possibility of using techniques like symbolic regression to learn explicit formulas representing (the whole or parts of) the physics accounted for by the corrective source term. If successful, such an approach may further increase the accuracy and generalizability of the CoSTA model. Additionally, we have discussed that the physicality of the corrective source term can be used to monitor the well-behavedness of the DDM component of the full CoSTA model. This facilitates in-built, fully automated sanity checks which can be used to avoid potentially dangerous model behaviour. In light of these observations, we believe that CoSTA can be a catalyst for DDM techniques to enter high-stakes applications presently reserved for pure PBM.

5.2. Further Work

In this thesis, we have chosen to use well-known, simple PBM and DDM techniques to demonstrate CoSTA, such that the technical details would not obscure the CoSTA framework itself. However, this also means that the power of CoSTA has not been fully explored in the present work. For example, we expect that the use of DDM techniques better suited for handling temporally correlated data will improve the performance of CoSTA-based models. In relation to this, it is important to investigate whether or not CoSTA maintains its edge over pure DDM when more advanced DDM techniques are considered. Among the many possible choices of DDM techniques, we highlight recurrent neural networks (RNNs) and temporal convolutional networks (TCNs) as being particularly interesting options, since these are designed specifically for use with temporally correlated data. The applicability of symbolic regression to CoSTA is also an interesting area of research.

A natural extension of the present work is to apply CoSTA to physical problems which are qualitatively different to the unsteady heat transfer problems that have been our primary focus. From the theoretical justification of CoSTA which we have presented, it follows that CoSTA is a valid approach for modelling *any* deterministic system. However, the ideal way of learning the corrective source term is not the same for all applications. Indeed, as we have noted, the approach we have used for heat transfer problems is not generally valid e.g. for systems that can develop discontinuities. From a practical point of view, it is important to discover such pitfalls, and to find general solutions for overcoming them. For the particular issue related to discontinuities, one possible fix is to adopt a neural network architecture that is better suited for handling time series, such as RNNs or TCNs.

At last, we highlight the importance of investigating the long-term stability of CoSTA-based models. As observed in our experiments, CoSTA is, to a certain extent, influenced by noise originating from the stochastic training process of its DNN. Currently, we have no theoretical framework to describe how such errors accumulate in time. In the most extreme consequence, this could result in divergent behaviour, even if the original PBM is convergent. In terms of trustworthiness, this is a significant issue which may delay the adoption of CoSTA in certain high-stakes applications. As rigorous frameworks for analysing the stability of pure PBM are already well-developed, a natural place to start is to investigate if these can be extended for use with hybrid models. For CoSTA models utilizing DNNs, these frameworks will likely have to be coupled with functional analysis, to take into account the influence of the DNN's output space on the overall behaviour of the hybrid model. In the current absence of a comprehensive theory on the stability of CoSTA-based models, it might also be worthwhile to develop heuristics to avoid the most severe consequences of error accumulation. One may for example apply filtering techniques to the corrective source term, such as to avoid divergent or otherwise unphysical behaviour. One may also explore the possibility of forcing the model to converge to the correct steady-state solution, if the steady-state solution is known. Overall, research into the long-term stability of CoSTA-based models would boost the trustworthiness of these models significantly. Consequently, CoSTA's relevance as an enabler for DDM to enter high-stakes application would be further increased.

Bibliography

Kenneth Omokhagbo Afebu, Yang Liu, Evangelos Papatheou, and Bingyong Guo. LSTM-based approach for predicting periodic motions of an impacting system via transient dynamics. *Neural Networks*, 140:49–64, 2021.

Shady E. Ahmed, Sk. Mashfiqur Rahman, Omer San, Adil Rasheed, and Ionel M. Navon. Memory embedded non-intrusive reduced order modeling of non-ergodic flows. *Physics of Fluids*, 31(12):126602, 2019.

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. doi: 10.1109/ACCESS.2018.2807385.

Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, Julien Brajard, and Patrick Gallinari. Learning dynamical systems from partial observations. *arXiv preprint arXiv:1902.11136*, 2019.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Zhongxin Bai and Xiao-Lei Zhang. Speaker recognition based on deep learning: An overview. *Neural Networks*, 2021.

Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS ONE*, 12(7):e0180944, 2017.

Ajay Bhattacharyya. Heat transfer analogies. Technical report, AB Atomenergi, 1965.

Chandrakant M. Bhumralkar. Numerical experiments on the computation of ground surface temperature in an atmospheric general circulation model. *Journal of Applied Meteorology and Climatology*, 14(7):1246–1258, 1975.

Sindre Stenen Blakseth. Data-driven correction methods for numerical solutions of the one-dimensional heat equation, 2021. Specialization project report for the NTNU-course TFY4510. Available online at `https://github.com/blakseth/TFY4510`.

Sindre Stenen Blakseth, Adil Rasheed, Trond Kvamsdal, and Omer San. Deep neural network enabled corrective source term approach to hybrid analysis and modeling. *arXiv preprint arXiv:2105.11521*, 2021.

Stefan Boschert and Roland Rosen. *Digital Twin—The Simulation Aspect*, pages 59–74. Springer International Publishing, 2016.

S. Brown, S. Martynov, and H. Mahgerefteh. Simulation of two-phase flow through ducts with discontinuous cross-section. *Computers & Fluids*, 120:46–56, 2015.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

Mengmeng Cai, Manisa Pipattanasomporn, and Saifur Rahman. Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. *Applied Energy*, 236:1078–1088, 2019.

Jie Chen, Guo-Qiang Zeng, Wuneng Zhou, Wei Du, and Kang-Di Lu. Wind speed forecasting using nonlinear-learning ensemble of deep learning time series prediction and extremal optimization. *Energy Conversion and Management*, 165:681–695, 2018.

Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv:2006.11287*, 2020.

Yuwei Fan and Lexing Ying. Solving electrical impedance tomography with deep learning. *Journal of Computational Physics*, 404:109119, 2020.

Joel Feldman. Solution of the wave equation by separation of variables, 2007. Available online at `https://secure.math.ubc.ca/~feldman/m267/separation.pdf`.

John B. Fraleigh. *A First Cource in Abstract Algebra.* Pearson Education Limited, 7th edition, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. Available online at `http://www.deeplearningbook.org`.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

David J. Griffiths. *Introduction to Electrodynamics.* Cambridge University Press, 4th edition, 2017.

Botros N. Hanna, Nam T. Dinh, Robert W. Youngblood, and Igor A. Bolotnov. Machine-learning based error prediction approach for coarse-grid computational fluid dynamics (CG-CFD). *Progress in Nuclear Energy*, 118:103140, 2020.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

P. C. Hemmer. *Kvantemekanikk.* Fagbokforlaget, 5th edition, 2015a.

P. C. Hemmer. *Termisk fysikk.* Fagbokforlaget, 2nd edition, 2015b.

Richard B. Hetnarski and M. Reza Eslami. *Thermal Stresses–Advanced Theory and Applications*, volume 158. Springer, 2nd edition, 2009.

Tim Hill, Marcus O'Connor, and William Remus. Neural network models for time series forecasts. *Management Science*, 42(7):1082–1092, 1996.

Jun-Ting Hsieh, Shengjia Zhao, Stephan Eismann, Lucia Mirabella, and Stefano Ermon. Learning neural PDE solvers with convergence guarantees. *arXiv preprint arXiv:1906.01200*, 2019.

Bruce Hunt and Nigel Cooke. Thermal calculations for bridge design. *Journal of the structural division*, 101(9):1763–1781, 1975.

J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

Ashu Jain and Avadhnam Madhav Kumar. Hybrid neural network models for hydrologic time series forecasting. *Applied Soft Computing*, 7(2):585–592, 2007.

Jürgen Jost. *Partial Differential Equations.* Springer-Verlag New York, 2nd edition, 2007.

Iebeling Kaastra and Milton Boyd. Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10(3):215–236, 1996.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Charles Kittel. *Introduction to Solid State Physics.* John Wiley & Sons, 8th edition, 2005.

Marc L. Kutner. *Astronomy: A Physical Perspective.* Cambridge University Press, 2nd edition, 2003.

Isaac E. Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

Judith Lean. Living with a variable sun. *Physics Today*, 58(6):32–38, 2005.

Judith L. Lean and David H. Rind. How natural and anthropogenic influences alter global and regional surface temperatures: 1889 to 2006. *Geophysical Research Letters*, 35(18), 2008.

Yu-Ju Lee, David Hall, Quan Liu, Wen-Wei Liao, and Ming-Chun Huang. Interpretable tropical cyclone intensity estimation using Dvorak-inspired machine learning techniques. *Engineering Applications of Artificial Intelligence*, 101:104233, 2021.

Randall J. LeVeque. *Finite-Volume Methods for Hyperbolic Problems.* Cambridge University Press, 1st edition, 2002.

Jan R. Lien and Gunnar Løvhøiden. *Generell fysikk for universiteter of høgskoler*, volume 1. Universitetsforlaget, 4th edition, 2015.

Egil Lillestøl, Ola Hunderi, and Jan R. Lien. *Generell fysikk for universiteter of høgskoler*, volume 2. Universitetsforlaget, 3rd edition, 2015.

Zhuo Liu, Chenhui Yao, Hang Yu, and Taihua Wu. Deep reinforcement learning with its application for lung cancer detection in medical Internet of Things. *Future Generation Computer Systems*, 97:1–9, 2019.

Alexandra Metallinou Log, Svend Tollak Munkejord, and Morten Hammer. HLLC-type methods for compressible two-phase flow in ducts with discontinuous area changes. *Computers & Fluids*, 2021.

Yungui Ma, Lu Lan, Wei Jiang, Fei Sun, and Sailing He. A transient thermal cloak experimentally realized through a rescaled diffusion equation with anisotropic thermal diffusivity. *NPG Asia Materials*, 5(11):e73–e73, 2013.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *30th International Conference on Machine Learning*, 2013.

Azad M. Madni, Carla C. Madni, and Scott D. Lucero. Leveraging digital twin technology in model-based systems engineering. *Systems*, 7(1):7, 2019.

Amal Mahmoud and Ammar Mohammed. A survey on deep learning for time-series forecasting. In *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*, pages 365–392. Springer, 2021.

Andreas Maier. Regularization–part 1, the bias-variance trade-off, 2020. Available online at `https://towardsdatascience.com/regularization-part-1-db408819b20f`.

R. Maulik, O. San, A. Rasheed, and P. Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, 2019. doi: 10.1017/jfm.2018.770.

Siddhartha Mishra. A machine learning framework for data driven acceleration of computations of differential equations. *arXiv preprint arXiv:1807.09519*, 2018.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1st edition, 1997.

Malte Müller, Mariken Homleid, Karl-Ivar Ivarsson, Morten A. Ø. Køltzow, Magnus Lindskog, Knut Helge Midtbø, Ulf Andrae, Trygve Aspelien, Lars Berggren, Dag Bjørge, Per Dahlgren, Jørn Kristiansen, Roger Randriamampianina, Martin Ridal, and Ole Vignes. Arome-metcoop: A nordic convective-scale operational weather prediction model. *Weather and Forecasting*, 32(2):609–627, 2017.

Arnaud Münch, Pablo Pedregal, and Francisco Periago. Relaxation of an optimal design problem for the heat equation. *Journal de Mathématiques Pures et Appliquées*, 89(3): 225–247, 2008.

Svend Tollak Munkejord and Morten Hammer. Depressurization of CO_2-rich mixtures in pipes: Two-phase flow modelling and comparison with experiments. *International Journal of Greenhouse Gas Control*, 37:398–411, 2015.

Svend Tollak Munkejord, Morten Hammer, and Sigurd W. Løvseth. CO_2 transport: Data and models—a review. *Applied Energy*, 169:499–523, 2016.

Thiruppudaimarudhur N. Narasimhan. Fourier's heat conduction equation: History, influence, and connections. *Reviews of Geophysics*, 37(1):151–172, 1999.

Elisa Negri, Luca Fumagalli, and Marco Macchi. A review of the roles of digital twin in CPS-based production systems. *Procedia Manufacturing*, 11:939–948, 2017.

Michael A. Nielsen. *Neural Networks and Deep Learning.* Determination Press, 2015. Available online at `http://neuralnetworksanddeeplearning.com/`.

Tor Nordam, Raymond Nepstad, Emma Litzler, and Johannes Röhrs. On the use of random walk schemes in oil spill modelling. *Marine Pollution Bulletin*, 146:631–638, 2019.

Zeydin Pala and Ramazan Atici. Forecasting sunspot time series using deep learning methods. *Solar Physics*, 294(5):1–14, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Jaideep Pathak, Mustafa Mustafa, Karthik Kashinath, Emmanuel Motheau, Thorsten Kurth, and Marcus Day. Using machine learning to augment coarse-grid computational fluid dynamics simulations. *arXiv preprint arXiv:2010.00072*, 2020.

Suraj Pawar, Shady E. Ahmed, Omer San, and Adil Rasheed. An evolve-then-correct reduced order model for hidden fluid dynamics. *Mathematics*, 8(4):570, 2020a.

Suraj Pawar, Shady E. Ahmed, Omer San, and Adil Rasheed. Data-driven recovery of hidden physics in reduced order modeling of fluid flows. *Physics of Fluids*, 32(3): 036602, 2020b.

Suraj Pawar, Shady E. Ahmed, Omer San, and Adil Rasheed. Hybrid analysis and modeling for next generation of digital twins. *arXiv preprint arXiv:2101.05908*, 2021a.

Suraj Pawar, Omer San, Burak Aksoylu, Adil Rasheed, and Trond Kvamsdal. Physics guided machine learning using simplified theories. *Physics of Fluids*, 33(1):011701, 2021b.

Francesca Pellicciotti, Marco Carenzo, Jakob Helbing, Stefan Rimkus, and Paolo Burlando. On the role of subsurface heat conduction in glacier energy-balance modelling. *Annals of Glaciology*, 50(50):16–24, 2009.

A. J. Pitman. The evolution of, and revolution in, land surface schemes designed for climate models. *International Journal of Climatology: A Journal of the Royal Meteorological Society*, 23(5):479–510, 2003.

G. P. Purja Pun, R. Batra, R. Ramprasad, and Y. Mishin. Physically informed artificial neural networks for atomistic modeling of materials. *Nature Communications*, 10(1): 1–10, 2019.

Alfio Quarteroni. *Numerical Models for Differential Problems*, volume 8. Springer-Verlag Italia, 2nd edition, 2014.

Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced Basis Methods For Partial Differential Equations: An Introduction*, volume 92. Springer, 2015.

Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357: 125–141, 2018.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. 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, 2019.

Rishikesh Ranade, Chris Hill, and Jay Pathak. DiscretizationNet: A machine-learning based solver for Navier–Stokes equations using finite volume discretization. *Computer Methods in Applied Mechanics and Engineering*, 378:113722, 2021.

Adil Rasheed, Omer San, and Trond Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, 8:21980–22012, 2020.

M. Risser, C. Vasile, T. Engel, B. Keith, and C. Muller. Numerical simulation of magnetocaloric system behaviour for an industrial application. *International Journal of Refrigeration*, 33(5):973–981, 2010.

Patrick J. Roache. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, 1998.

Patrick J. Roache. Code verification by the Method of Manufactured Solutions. *Journal of Fluids Engineering*, 124(1):4–10, 2002.

Thomas Roger, Calum Maitland, Kali Wilson, Niclas Westerberg, David Vocke, Ewan M. Wright, and Daniele Faccio. Optical analogues of the Newton–Schrödinger equation and boson star evolution. *Nature Communications*, 7(1):1–8, 2016.

Roland Rosen, Georg Von Wichert, George Lo, and Kurt D. Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 48(3):567–572, 2015.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13, 2019.

Omer San, Adil Rasheed, and Trond Kvamsdal. Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution. *GAMM-Mitteilungen*, 44:e202100007, 2021.

Benjamin Schleich, Nabil Anwer, Luc Mathieu, and Sandro Wartzack. Shaping the digital twin for design and production engineering. *CIRP Annals*, 66(1):141–144, 2017.

Mike Shafto, Mike Conroy, Rich Doyle, Ed Glaessgen, Chris Kemp, Jacqueline LeMoigne, and Lui Wang. Modeling, simulation, information technology & processing roadmap. *National Aeronautics and Space Administration*, 2012.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Justin Sirignano, Jonathan F. MacArt, and Jonathan B. Freund. DPM: A deep learning PDE augmentation method with application to large-eddy simulation. *Journal of Computational Physics*, 423:109811, 2020.

John W. Slater. Examining spatial (grid) convergence, 2021. Available online at `https://www.grc.nasa.gov/www/wind/valid/tutorial/spatconv.html`.

Joseph Smagorinsky. General circulation experiments with the primitive equations. *Monthly Weather Review*, 91(3):99–164, 1963.

Michael Stanko and Markus Stommel. Digital twin of the polyurethane rotational moulding process. In *Advances in Polymer Processing 2020*, pages 324–335. Springer, 2020.

Steven H. Strogatz. *Nonlinear Dynamics and Chaos.* Westview Press, 2nd edition, 2015.

Jian Sun, Zhan Niu, Kristopher A. Innanen, Junxiao Li, and Daniel O. Trad. A theory-guided deep-learning formulation and optimization of seismic waveform inversion. *Geophysics*, 85(2):R87–R99, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

John C. Tannehill, Dale A. Anderson, and Richard A. Pletcher. *Computational Fluid Mechanics and Heat Transfer.* Taylor & Francis, 2nd edition, 1997.

Fei Tao, Jiangfeng Cheng, Qinglin Qi, Meng Zhang, He Zhang, and Fangyuan Sui. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, 94(9):3563–3576, 2018.

J. W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*, volume 22. Springer Science+Business Media, LLC, 1st edition, 1995.

Michael L. Thompson and Mark A. Kramer. Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 40(8):1328–1340, 1994.

E. F. Toro. NUMERICA: A library of source codes for teaching, research and applications, 1999. Available online at `https://eleuteriotoro.com/software`.

Duy Tan Tran, Haakon Robinson, Adil Rasheed, Omer San, Mandar Tabib, and Trond Kvamsdal. Gans enabled super-resolution reconstruction of wind field. In *Journal of Physics: Conference Series*, volume 1669, page 012029. IOP Publishing, 2020.

Alex Y. Tuan and G. Q. Shang. Vibration control in a 101-storey building using a tuned mass damper. *Journal of Applied Science and Engineering*, 17:141–156, 2014.

Harsha Vaddireddy, Adil Rasheed, Anne E. Staples, and Omer San. Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data. *Physics of Fluids*, 32(1):015113, 2020.

Nico Van der Wijst. *Finance: A Quantitative Introduction.* Cambridge University Press, 1st edition, 2013.

H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics.* Longman Scientific & Technical, 1st edition, 1995.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, 8(8):876, 2019.

Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, and Themistoklis Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS ONE*, 13(5):e0197704, 2018.

Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on DNS data. *Physical Review Fluids*, 2(3):034603, 2017.

Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 2020.

Jin-Long Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 3(7):074602, 2018.

You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. tempoGAN: a temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Liu Yang, Dongkun Zhang, and George Em Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *SIAM Journal on Scientific Computing*, 42(1):A292–A317, 2020.

Abdelhafid Zeroual, Fouzi Harrou, Abdelkader Dairi, and Ying Sun. Deep learning methods for forecasting COVID-19 time-series data: A comparative study. *Chaos, Solitons & Fractals*, 140:110121, 2020.

G. Peter Zhang. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159–175, 2003.

A. Correcting Fluid Flow Problems Using CoSTA

In this chapter, we study the application of CoSTA to a physical problem that is qualitatively different to the unsteady heat transfer problems studied in the main text. More specifically, we now focus on applying CoSTA to fluid flow problems described by the Euler equations. These equations are commonly used in simulations of fluid flow in pipelines (Brown et al., 2015; Munkejord and Hammer, 2015; Log et al., 2021), and are therefore relevant for e.g. hydrocarbon transport and carbon capture and storage. They can be derived from the Navier–Stokes equations by neglecting second-order derivatives, and constitute a system of so-called *hyperbolic* equations. Hyperbolic equations are fundamentally different from *parabolic* equations like the unsteady heat equation in the sense that hyperbolic equations permit discontinuous solutions, while parabolic equations do not. This makes the phenomenon of numerical diffusion much more pronounced for hyperbolic equations. Simply put, numerical diffusion refers to artificial smoothening that occurs in numerical solvers. For example, it can result in a discontinuity in the true solution of a hyperbolic governing equation being replaced by a smooth transition in the corresponding numerical solution, thereby reducing accuracy. The main goal of this chapter is to investigate if CoSTA can alleviate the issue of numerical diffusion in numerical solutions of hyperbolic governing equations.

A.1. Physics-Based Modelling of Fluid Flow

Here, we describe a physics-based model for fluid flow governed by the Euler equation. Our description is, in its entirety, based on the excellent textbook by LeVeque (2002). A brief introduction to the Euler equations and its solutions for so-called Riemann problems is given in Section A.1.1. Subsequently, we present a method for solving the Euler equations numerically in Section A.1.2.

A.1.1. The Euler Equations

The 1D Euler equations describe the mass balance, momentum balance and energy balance for closed systems with constant cross-section, no source terms and no viscous stresses, as expressed mathematically by the three equations below

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho v}{\partial x} = 0, \tag{A.1}$$

$$\frac{\partial \rho v}{\partial t} + \frac{\partial (\rho v^2 + p)}{\partial x} = 0, \tag{A.2}$$

$$\frac{\partial E}{\partial t} + \frac{\partial (E + p) v}{\partial x} = 0. \tag{A.3}$$

Here, v denotes velocity in the x-direction, p denotes pressure and E denotes total energy per unit volume, while ρ, x and t retain their meaning from the main text. The energy E is given as the sum of the kinetic energy $\frac{1}{2}\rho v^2$ and the internal energy ρe, where

e denotes specific internal energy. The 1D Euler equations can also be written more compactly as

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} = \boldsymbol{0}, \tag{A.4}$$

where

$$\boldsymbol{U} = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = \begin{pmatrix} \rho \\ \rho v \\ E \end{pmatrix} \tag{A.5}$$

is known as the vector of conserved variables and

$$\text{and} \quad \boldsymbol{F}(\boldsymbol{U}) = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} \rho v \\ \rho v^2 + p \\ (E+p)v \end{pmatrix} \tag{A.6}$$

is known as the flux vector. The 1D Euler equations do not constitute a closed system of equations on their own; we must also specify a so-called equation of state relating pressure, density and temperature. For our purposes it is sufficient to consider the simple *ideal gas* equation of state

$$p = \rho c_V (\gamma - 1) T, \tag{A.7}$$

where $\gamma = c_p/c_V$ is the ratio between the specific heat capacity at constant pressure, c_p, and the specific heat capacity at constant volume, c_V.

In applications, we are often not only interested in the conserved variables $\boldsymbol{U}$, but also in variables such as the pressure p, the velocity v, and the temperature T. For the ideal gas equation of state, the internal energy e can be expressed as

$$e = E - \frac{1}{2}v^2 = \frac{U_3}{U_1} - \frac{1}{2}\left(\frac{U_2}{U_1}\right)^2 \tag{A.8}$$

such that the variables p, v and T can be computed from the conserved variables using the following relations:

$$p = (\gamma - 1)\rho e = (\gamma - 1)\left(U_3 - \frac{U_2^2}{U_1}\right), \tag{A.9}$$

$$v = \frac{\rho v}{\rho} = \frac{U_2}{U_1}, \tag{A.10}$$

$$T = \frac{e}{c_V} = \frac{1}{c_V}\left(\frac{U_3}{U_1} - \frac{1}{2}\left(\frac{U_2}{U_1}\right)^2\right). \tag{A.11}$$

In the study of fluid flow governed by the 1D Euler equations, *Riemann problems* is a particularly important class of problems. Riemann problems are characterized by a piecewise constant initial condition consisting of a left state $\boldsymbol{U}_\mathrm{L}$ and a right state $\boldsymbol{U}_\mathrm{R}$ separated by a discontinuity. For the 1D Euler equations, all Riemann problems can be solved exactly, and this has two major benefits. The first benefit is that we can then compare numerical solutions of Riemann problems to the corresponding exact solutions, thereby establishing the accuracy of the numerical solutions. The second benefit is that the behaviour of solutions of the 1D Euler equations for Riemann problems are well-understood. In particular, it is known that the solution of any Riemann problem is characterized by the presence of four constant states $\boldsymbol{U}_\mathrm{L}$, $\boldsymbol{U}_\mathrm{L}^*$, $\boldsymbol{U}_\mathrm{R}^*$ and $\boldsymbol{U}_\mathrm{R}$, where $\boldsymbol{U}_\mathrm{L}$ and

$\boldsymbol{U}_{\mathrm{R}}$ are the initial states. The states $\boldsymbol{U}_{\mathrm{L}}$ and $\boldsymbol{U}_{\mathrm{L}}^*$ are either separated by a rarefaction wave (a finite region where $\boldsymbol{U}$ transitions smoothly between the two states) or a shock (a discontinuous transition between the two states). Similarly, the states $\boldsymbol{U}_{\mathrm{R}}$ and $\boldsymbol{U}_{\mathrm{R}}^*$ are also separated by either a rarefaction wave or a shock. Lastly, the states $\boldsymbol{U}_{\mathrm{L}}^*$ and $\boldsymbol{U}_{\mathrm{R}}^*$ are separated by a so-called contact discontinuity across which certain properties (e.g. ρ) are discontinuous while others (e.g. v and p) are constant. Note that if $\boldsymbol{U}_{\mathrm{L}}^* = \boldsymbol{U}_{\mathrm{L}}$ and $\boldsymbol{U}_{\mathrm{R}}^* = \boldsymbol{U}_{\mathrm{R}}$, we have a Riemann problem where the solution contains an isolated contact discontinuity. This is the class of Riemann problems we will study in our numerical experiment (cf. Section A.2).

A.1.2. The Lax–Friedrichs Method

The Lax–Friedrichs Method (LxF) is a widely used method for solving the 1D Euler equations. It is well-renowned for its robustness, but also suffers from severe numerical dissipation, which results in unsatisfactory accuracy in many applications. However, for our purposes, the dissipative nature of LxF is precisely the reason we choose to use it; since its dissipative behaviour is well-understood, we know exactly what errors to expect. It is then easy to observe if CoSTA is successful in correcting these errors.

Using the spatial and temporal discretizations defined in Section 2.2.3, LxF can be written on the following form:

$$\boldsymbol{U}_j^{n+1} = \boldsymbol{U}_j^n - \frac{\Delta t}{\Delta x}\left(\boldsymbol{\mathcal{F}}_{j+1/2}^n - \boldsymbol{\mathcal{F}}_{j-1/2}^n\right), \quad j = 1, \ldots, N_j \tag{A.12}$$

where

$$\boldsymbol{\mathcal{F}}_{j+1/2}^n = \frac{1}{2}\left(\boldsymbol{F}(\boldsymbol{U}_j^n) + \boldsymbol{F}(\boldsymbol{U}_{j+1}^n) - \frac{\Delta x}{\Delta t}(\boldsymbol{U}_{j+1}^n - \boldsymbol{U}_j^n)\right), \quad j = 1, \ldots, N_j - 1, \tag{A.13}$$

and $\boldsymbol{\mathcal{F}}_{j-1/2}^n$ is defined analogously for $j = 2, \ldots, N_j$. $\boldsymbol{\mathcal{F}}$ is known as the numerical flux vector, and has presently been defined at all interior cell faces. To define the numerical flux vector at the boundary cell faces $j = 1/2$ and $j = N_j + 1/2$, we use extrapolation boundary conditions. That is, we assume $\boldsymbol{U}_0^n = \boldsymbol{U}_1^n$ and $\boldsymbol{U}_{N_j+1}^n = \boldsymbol{U}_{N_j}^n$. This yields

$$\boldsymbol{\mathcal{F}}_{1/2}^n = \boldsymbol{F}(\boldsymbol{U}_1^n) \quad \text{and} \quad \boldsymbol{\mathcal{F}}_{N_j+1/2}^n = \boldsymbol{F}(\boldsymbol{U}_{N_j}^n). \tag{A.14}$$

Unlike the Implicit Euler FVM considered in Section 2.2.3, LxF is not stable for all choices of Δx and Δt. Indeed, LxF is only stable if our chosen Δx and Δt satisfy the so-called CFL-condition:

$$\frac{\Delta x}{\Delta t} \geq \max_{j \in [1, N_j]} \left\{|v_j| + c_j\right\}. \tag{A.15}$$

Here, c is the speed of sound. For the ideal gas equation of state, it is given as

$$c_j = \sqrt{(\gamma - 1)\,\gamma c_V T_j}, \quad j = 1, \ldots, N_j. \tag{A.16}$$

A.2. Numerical Experiment

In this section, we consider a single numerical experiment where we attempt to use CoSTA to reduce numerical dissipation in numerical solutions of the 1D Euler equations computed using LxF.

A.2.1. Experimental Setup and Method

In the present experiment, we consider a series of Riemann problems where the true solution consists of a constant left state $\boldsymbol{U}_\mathrm{L}$ and a constant right state $\boldsymbol{U}_\mathrm{R}$ separated by a contact discontinuity (CD). The velocity of this CD is different for the different Riemann problems, and we parametrize its velocity (which is equal to the velocity v of both the left state and the right state) by a parameter α. The right and left states considered in our experiment are given by

$$\boldsymbol{U}_\mathrm{L} = \begin{pmatrix} \rho_\mathrm{L} \\ \rho_\mathrm{L} v \\ E_\mathrm{L} \end{pmatrix} = \begin{pmatrix} 1.4 \\ 0.14\alpha \\ 2.5 + 0.007\alpha^2 \end{pmatrix}, \quad \boldsymbol{U}_\mathrm{R} = \begin{pmatrix} \rho_\mathrm{R} \\ \rho_\mathrm{R} v \\ E_\mathrm{R} \end{pmatrix} = \begin{pmatrix} 1.0 \\ 0.1\alpha \\ 2.5 + 0.005\alpha^2 \end{pmatrix}. \tag{A.17}$$

A complete overview of our Riemann problem configurations can be found in Table A.1. We use LxF with extrapolation boundary conditions as our baseline PBM for solving these Riemann problems. Additionally, we use CoSTA to define a HAM model based on LxF. More specifically, we augment Equation (A.12) with a DNN-generated corrective source term $\hat{\boldsymbol{\sigma}}_\mathrm{NN}$ such that we obtain

$$\boldsymbol{U}_j^{n+1} = \boldsymbol{U}_j^n - \frac{\Delta t}{\Delta x}\left(\boldsymbol{\mathcal{F}}_{j+1/2}^n - \boldsymbol{\mathcal{F}}_{j-1/2}^n\right) + (\hat{\boldsymbol{\sigma}}_\mathrm{NN}^{n+1})_j, \quad j = 1, \ldots, N_j. \tag{A.18}$$

Note that while $\hat{\boldsymbol{\sigma}}_\mathrm{NN}$ was a vector when we considered the Implicit Euler FVM in the main text, it is now a rank-2 tensor in $\mathbb{R}^3 \times \mathbb{R}^{N_j}$. We will shortly explain how we train the DNN that generates $\hat{\boldsymbol{\sigma}}_\mathrm{NN}$ in our experiments. However, as this explanation will be somewhat lengthy, we first mention which α-values we have used to define our series of Riemann problems.

As in the experiments of Chapter 4, we consider 22 different α-values split into three sets, $\mathcal{A}_\text{train}$, $\mathcal{A}_\text{val}$ and $\mathcal{A}_\text{test}$. We choose $\mathcal{A}_\text{train}$ as the roots of the 16th degree Legendre polynomial on the interval $[0, 1]$. Furthermore, we choose $\mathcal{A}_\text{val} = \{0.25, 0.75\}$ and $\mathcal{A}_\text{test} = \{0.1, 0.5, 0.9, 1.5\}$. The 16 α-values in $\mathcal{A}_\text{train}$ are used for DNN training, the 2 α-values in $\mathcal{A}_\text{val}$ are used for DNN validation, and the 4 α-values in $\mathcal{A}_\text{test}$ are used for model evaluation (testing). Since we are considering Riemann problems, we can obtain exact solutions $\boldsymbol{U}_\text{ref}$ for all 22 α-values using e.g. the solver found in the NUMERICA library (Toro, 1999).

We will now discuss how to train a DNN to generate the corrective source term $\hat{\boldsymbol{\sigma}}_\mathrm{NN}$ used in Equation (A.18). The first thing we must do is to establish the DNN training targets. Given an an exact solution $\boldsymbol{U}_\text{ref}$ evaluated at grid nodes $x_1, \ldots, x_{N_j}$ and time levels n and $n+1$, we define (the columns of) the ideal corrective source term as

$$\hat{\boldsymbol{\sigma}}_j^{n+1} = (\boldsymbol{U}_\text{ref}^{n+1})_j - (\boldsymbol{U}_\text{ref}^n)_j - \frac{\Delta t}{\Delta x}\left(\boldsymbol{\mathcal{F}}_{j+1/2}^n - \boldsymbol{\mathcal{F}}_{j-1/2}^n\right), \quad j = 1, \ldots, N_j, \tag{A.19}$$

where the numerical flux vector is also calculated using the exact solution $\boldsymbol{U}_\text{ref}$. With the explicit formula for the ideal corrective source term given above, it is tempting to follow our approach from Section 3.3 and train the DNN to learn the following mapping

$$\begin{aligned} \mathrm{DNN}_\sigma : \mathbb{R}^3 \times \mathbb{R}^{N_j} &\to \mathbb{R}^3 \times \mathbb{R}^{N_j} \quad \text{such that} \quad \hat{\boldsymbol{\sigma}}_\mathrm{NN}^{n+1} = \hat{\boldsymbol{\sigma}}^{n+1}, \\ \widetilde{\boldsymbol{U}}_\mathrm{h}^{n+1} &\mapsto \hat{\boldsymbol{\sigma}}_\mathrm{NN}^{n+1} \end{aligned} \tag{A.20}$$

where $\widetilde{\boldsymbol{U}}_\mathrm{h}^{n+1}$ is an approximation of $\boldsymbol{U}^{n+1}$ calculated using Equation (A.12) with $\boldsymbol{U}_\mathrm{h}^n$ inserted in the place of $\boldsymbol{U}^n$. However, for the present Riemann problems, such an

Table A.1.: Configuration of Riemann problems. All quantities given in their respective SI unit. Quantities not listed can be computed using those that are listed. Note that the choices of N_j and Δt are such that the CFL condition (A.15) is satisfied for all experiments considered herein.

Quantity	Value
Left-state density, ρ_{L}	1.4
Left-state velocity, u_{L}	0.1α
Left-state pressure, p_{L}	1.0
Right-state density, ρ_{R}	1.0
Right-state velocity, u_{R}	0.1α
Right-state pressure, p_{R}	1.0
Temporal domain, $[t_0, t_{\mathrm{end}}]$	[0.0, 1.4]
Spatial domain, $[x_a, x_b]$	[0.0, 1.0]
Initial location of CD, x_{CD}	0.5
Specific heat capacity at constant volume, c_V	2.5
Ratio of heat capacities, γ	1.4
Number of grid cells, N_j	100
Time step, Δt	5e-3

approach will fail. The reason is simply that the relation defined in Equation (A.20) does not generally satisfy the definition of a map (Fraleigh, 2014, page 4). The problem is that Equation (A.20) relates predictors $\widetilde{\boldsymbol{U}}_{\mathrm{h}}^{n+1}$ that are equal to each other with corrective source terms $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}^{n+1}$ that are different from each other. That is, some elements of the input space are related to multiple elements of the output space, and this is prohibited by the definition of a map. The key to realizing why this occurs, is to study the CFL-condition in greater detail. In all our problems, v is constant and larger than zero, such that the CFL-condition can be rewritten as

$$\Delta x \geq \max_{j\in[1,N_j]} \{v + c_j\}\, \Delta t. \tag{A.21}$$

We know that the speed of sound is always strictly positive, such that v is strictly smaller than $\max\limits_{j\in[1,N_j]} \{v + c_j\}$. We also know that the rarefaction wave travels at velocity v for all problems considered here. Thus, the distance $v\Delta t$ travelled by the rarefaction wave during a time step Δt is always strictly smaller than the grid spacing Δx due to the CFL condition. This means that there always exists some time level $n' \geq 0$ and some grid node $x_{j'}$ such that the following holds:

1. The true location of the contact discontinuity at time level n' is in the interval $[x_{j'-1}, x_{j'})$.

2. The true locations of the contact discontinuity at time levels $n'+1$ and $n'+2$ are in the interval $[x_{j'}, x_{j'+1})$.

For the sampled reference solutions at time levels n', $n'+1$ and $n'+2$, the above implies $\boldsymbol{U}_{\mathrm{ref}}^{n'} \neq \boldsymbol{U}_{\mathrm{ref}}^{n'+1} = \boldsymbol{U}_{\mathrm{ref}}^{n'+2}$. This further implies that $\widetilde{\boldsymbol{U}}_{\mathrm{h}}^{n'} \neq \widetilde{\boldsymbol{U}}_{\mathrm{h}}^{n'+1} = \widetilde{\boldsymbol{U}}_{\mathrm{h}}^{n'+2}$. Simultaneously, it also implies that Equation (A.19) yields $\hat{\boldsymbol{\sigma}}^{n'+1} \neq \hat{\boldsymbol{\sigma}}^{n'+2}$. In summary, we then have $\widetilde{\boldsymbol{U}}_{\mathrm{h}}^{n'+1} = \widetilde{\boldsymbol{U}}_{\mathrm{h}}^{n'+2}$ and $\hat{\boldsymbol{\sigma}}^{n'+1} \neq \hat{\boldsymbol{\sigma}}^{n'+2}$, which means that Equation (A.20) does not define a map. Through analogous arguments, one can also show that using another sampled $\boldsymbol{U}$ (e.g. $\boldsymbol{U}^n_{\mathrm{ref}}$) as DNN input does not remedy the problem; no relation like the one defined in Equation (A.20), where only data from a single time level is used as DNN input, can ever be a well-defined map for the present class of problems. For that reason, attempts at training DNNs to approximate such relations will necessarily be unsuccessful.

A naive attempt at fixing the problem outlined above would be to use DNN input sampled at several time levels. While this is conceivably doable when Δx and Δt are fixed and bounds of c can be determined, it is still poses some problems of its own. First of all, if v is such that data from three time levels must be included for the DNN input to be unique, this means we cannot straight-forwardly apply any correction for the first time step. This error will then be propagated through to later time levels, and the DNN-generated corrective source term will never be able to account for or correct this error. Note also that the severity of this problem increases if data from an even larger number of time levels must be used as DNN input. Another problem is that this approach is highly inefficient in its utilization of data. Consider for example our test scenario where $\alpha = 0.1$. In this scenario, it will take 200 time steps for the CD to propagate a distance of Δx, which means that we must use data sampled at 200 time levels as DNN input for the learning problem of the DNN to be well-posed. This is clearly an undesirable use of computational resources.

To define a well-posed learning problem for the DNN, we turn to a manual inspection of the true corrective source terms calculated from the training data using Equation (A.19). It is then immediately clear that the corrective source terms are very sparse, containing only two non-zero entries per conserved variable. One will also quickly realize that the peak heights depend on only two factors: 1) the parameter α, and 2) whether or not the CD moved from one grid cell to another during the previous time step. Furthermore, we empirically find that the peak heights for a time step where the CD moved from one grid cell to another can be calculated using the peak heights for a time step where the CD stayed within a single grid cell, and vice versa. Thus, if we can predict the peak heights at a single time level given α, we can reconstruct the corrective source term for that α at all grid nodes and time levels. That is, given a single parameter α, we want to predict a total of six peak heights. To make this prediction task easier, through careful observation of the training data, the author was able to identify an explicit formula[1] for the peaks in the first row of $\hat{\boldsymbol{\sigma}}$. For predicting the final 4 peak heights, we resort to using a DNN. Upon collecting the unknown peak heights in a single vector denoted $\boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3, \xi_4]$, the ideal mapping for the DNN can be written as

$$\begin{aligned} \mathrm{DNN}_\xi : \mathbb{R} &\to \mathbb{R}^4 \quad \text{such that} \quad \boldsymbol{\xi}_{\mathrm{NN}} = \boldsymbol{\xi}. \\ \alpha &\mapsto \boldsymbol{\xi}_{\mathrm{NN}} \end{aligned} \tag{A.22}$$

Note that we now use the subscript $_\xi$ to indicate that the DNN will be trained to predict the corrective source term peak heights $\boldsymbol{\xi}$, not the corrective source term $\hat{\boldsymbol{\sigma}}$ itself. Our predicted corrective source term $\hat{\boldsymbol{\sigma}}_{\mathrm{NN}}$ will be computed from the DNN-predicted peak heights $\boldsymbol{\xi}_{\mathrm{NN}}$ using the relations between $\hat{\boldsymbol{\sigma}}$ and $\boldsymbol{\xi}$ that were identified manually.

We have now finally obtained a well-defined mapping which we can train a DNN to approximate. To this end, we use the same training procedure as described in Section 4.1. We also continue to use the same fully-connected DNN architecture with the same hyperparameters (cf. Table 4.4), except for two small modifications. First of all, the input and output dimensionality must be altered to accommodate the mapping (A.22). Furthermore, in an effort to reduce overfitting we have reduced the layer width to 25. We

[1]It's precise mathematical form does not add value to the present discussion, so it is not written here.

also follow the procedure outlined in Section 4.1 for model evaluation, with the exception that we do not include any pure DDM in the present experiment.

A.2.2. Results

In Figure A.1, we present the temporal development of the ℓ_2-errors $\|\boldsymbol{U}_{\mathrm{p}}^n - \boldsymbol{U}_{\mathrm{ref}}^n\|_2/\|\boldsymbol{U}_{\mathrm{ref}}^n\|_2$ and $\|\boldsymbol{U}_{\mathrm{h}}^n - \boldsymbol{U}_{\mathrm{ref}}^n\|_2/\|\boldsymbol{U}_{\mathrm{ref}}^n\|_2$ of the PBM predictions $\boldsymbol{U}_{\mathrm{p}}^n$ and the HAM predictions $\boldsymbol{U}_{\mathrm{h}}^n$. From the figure, it is clear that the HAM drastically outperforms PBM. In fact, the use of the DNN-generated corrective source term in HAM has lowered the relative ℓ_2-error by at least two orders of magnitude in all interpolation scenarios when comparing to stand-alone PBM. The performance difference is particularly clear for $\alpha = 0.5$, which lies in the middle of the domain from which the α-values in $\mathcal{A}_{\mathrm{train}}$ were drawn. The accuracy of HAM is slightly degraded in the extrapolation scenario $\alpha = 1.5$, but its accuracy is still far superior to that of the PBM.

The reader is made aware that the error spikes in Figure A.1d are caused by rounding errors resulting in erroneous predictions of the location of the contact discontinuity, which in turn cause the corrective source term to be applied at the wrong locations. In cases where the exact location of the contact discontinuity is precisely at a grid node, LxF must still define a state at the grid node, and rounding error can influence if LxF chooses the right state or the left state at the grid node. This also applies to the exact solver in the NUMERICA library, and the large error spikes result from NUMERICA and LxF being influenced differently by the rounding errors, thereby choosing different states at the location of the CD.

In Figures A.2–A.5, we display the pressure p, velocity v, temperature T and density ρ as predicted by PBM and HAM at the final time $t_{\mathrm{end}} = 1.4\,\mathrm{s}$. From the T- and ρ-profiles, one can observe that the prominent numerical diffusion of LxF has been significantly reduced by the CoSTA-based HAM model. Indeed, the HAM-predicted T- and ρ-profiles appear truly discontinuous, while significant smoothening is present in the corresponding PBM-predicted profiles. Thus, HAM is successful in reducing numerical diffusion. However, as can be seen from the p- and v-profiles, the HAM predictions are more noisy than the PBM predictions. While the scaling of the vertical axis makes the noise look more prominent than it really is, it can still be of significance. As the long-term stability of CoSTA-based models has yet to be guaranteed, as discussed in the main text, one cannot rule out that this noise will be amplified over time, thereby diminishing predictive accuracy over large temporal domains. Another issue is that the noise may result in the predicted state being unphysical. In certain applications, even solutions that are only marginally unphysical (meaning they are very similar to physical solutions) may present serious problems. For example, if the numerical solver is coupled with a thermodynamics library, the thermodynamics library may fail when presented with any unphysical prediction, regardless of the severity of the unphysicality. This was not a problem we had to consider for unsteady heat transfer, as no single temperature profile can be deemed unphysical and thereby crash a heat transfer simulation.

A.2.3. Discussion

The experimental results presented here show that CoSTA can be used to reduce numerical dissipation in numerical solutions of the 1D Euler equations. More broadly, the results also show that CoSTA is applicable to systems with hyperbolic governing equations. This drastically widens the potential impact of CoSTA, as many important PDEs

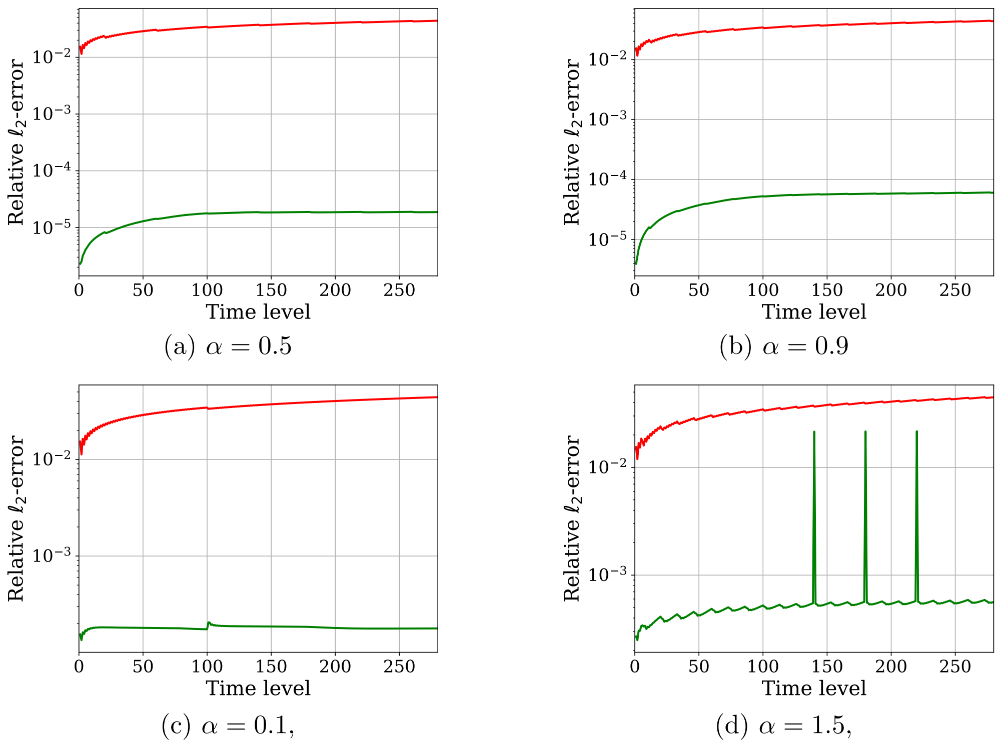


Figure A.1.: Euler equations: Relative ℓ_2-errors for all $\alpha \in \mathcal{A}_{\text{test}}$ (—PBM, —HAM).

and PDE systems are hyperbolic, including the wave equation and the inviscid Burgers' equation in addition to the Euler equations studied herein.

With the above said, it is clear that the particular example studied in this chapter has limited value in real-world applications, because we have restricted ourselves to study a problem for which the exact solution is readily available. The DNN training routine used here is purely based on empirical observations of the training set for Riemann problems with isolated CDs. Therefore, it is unclear to what extent similar procedures would be successful in more complex scenarios. In particular, we do not presently know how interaction between the contact discontinuity and any possible shocks or rarefaction waves will affect the nature of the corrective source term. If corrections add linearly, a good corrective model can possibly be developed by studying rarefaction waves, contact discontinuities and shocks independently, and then later combining the results into a more general model. However, if they prove to be non-linear, a different approach must probably be taken. It also remains to investigate how the presence of non-conservative terms will affect the dynamics of CoSTA-based HAM models for the Euler equations. Still, corrective source terms defined using Equation (A.19) are always uniquely defined by x, t and α (since $\boldsymbol{U}_{\text{ref}}$ is uniquely defined by x, t and α). Thus, an appropriate DNN mapping always *exists* – the difficult part is finding one that can be learnt efficiently. In that respect, the experiment presented in this chapter conveniently illustrates some of the challenges one might face when developing a CoSTA-based HAM model. It also motivates further research concerning the applicability of CoSTA to hyperbolic problems.

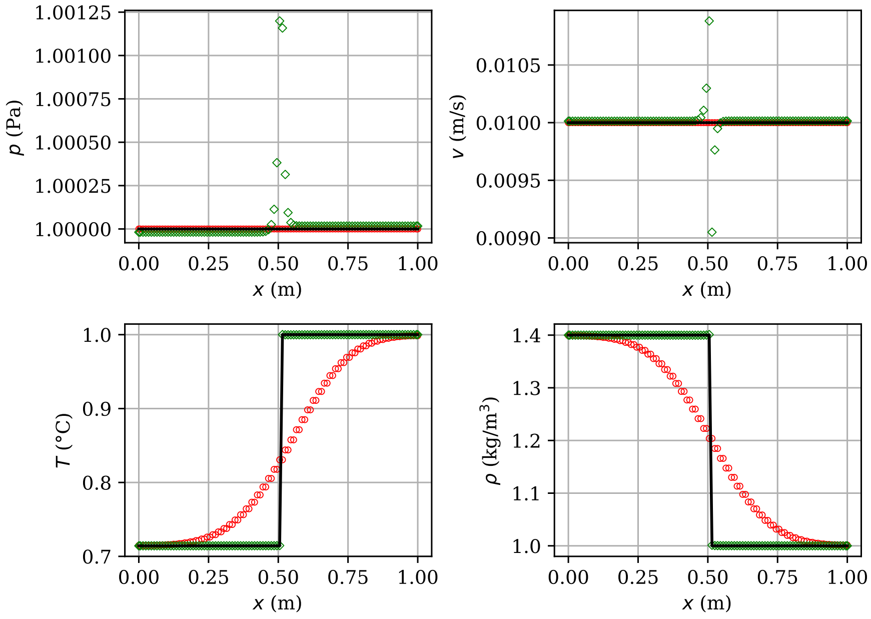


Figure A.2.: Euler equations: Predicted pressure, velocity, temperature and density profiles at the final time level for $\alpha = 0.1$ (○ PBM, ◇ HAM).

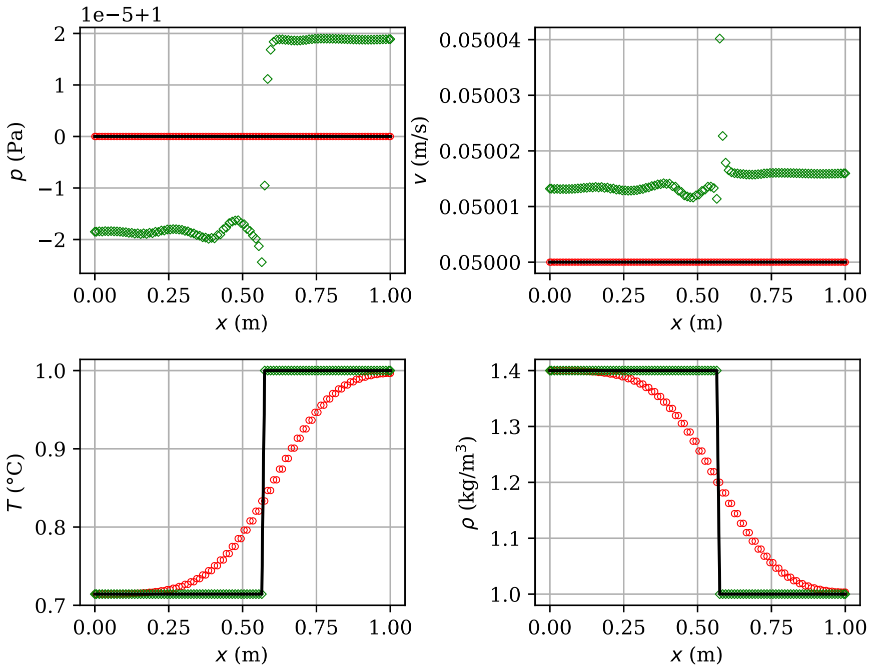


Figure A.3.: Euler equations: Predicted pressure, velocity, temperature and density profiles at the final time level for $\alpha = 0.5$ (○ PBM, ◇ HAM).

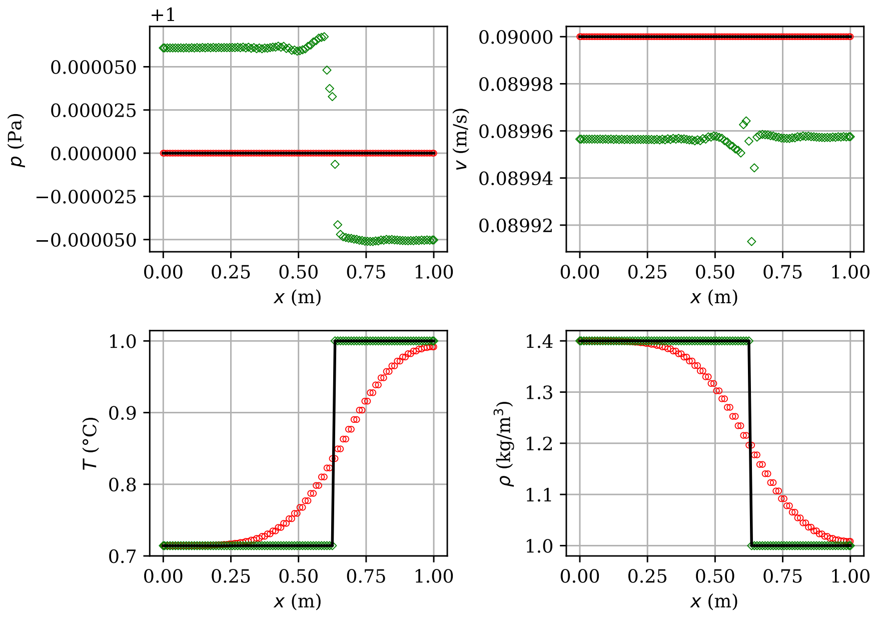


Figure A.4.: Euler equations: Predicted pressure, velocity, temperature and density profiles at the final time level for $\alpha = 0.9$ (○ PBM, ◇ HAM).

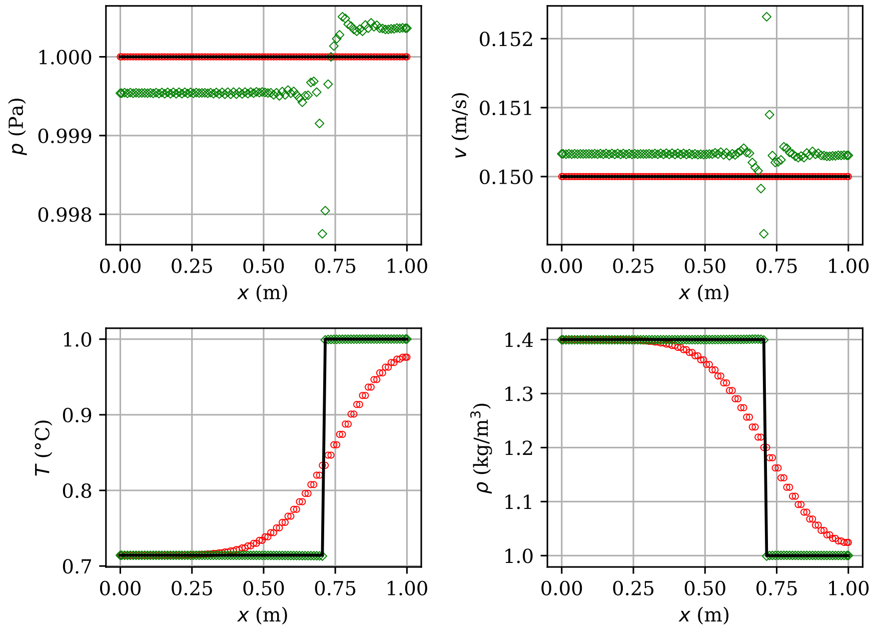


Figure A.5.: Euler equations: Predicted pressure, velocity, temperature and density profiles at the final time level for $\alpha = 1.5$ (○ PBM, ◇ HAM).

B. Correcting Boundary Conditions Using CoSTA

The purpose of this chapter is to show how to correct the boundary conditions (BCs) of the one-dimensional (1D) Implicit Euler Finite Volume Method (FVM) using the Corrective Source Term Approach (CoSTA).[1] As shown in Section 2.2.3, the 1D Implicit Euler FVM can be written on the form

$$\mathbb{A}\boldsymbol{T}^{n+1} = \boldsymbol{b}\left(\boldsymbol{T}^{n}\right), \tag{B.1}$$

where $\boldsymbol{T} = [T_1, \ldots, T_{N_j}]$ describes the temperature at all N_j *interior* grid nodes (which we denote $x_1, \ldots, x_{N_j}$), and the superscripts denote the time level at which the grid node temperatures are evaluated. Furthermore, $\mathbb{A}$ is a tridiagonal matrix whose non-zero elements are

$$\begin{aligned}
a_{j,j} &= 1 + \kappa\frac{\Delta t}{\Delta x}\left(\frac{1}{\Delta x_{j+1/2}} + \frac{1}{\Delta x_{j-1/2}}\right), && j = 1, \ldots, N_j,\\
a_{j,j+1} &= -\kappa\frac{\Delta t}{\Delta x}\frac{1}{\Delta x_{j+1/2}}, && j = 1, \ldots, N_j - 1,\\
a_{j-1,j} &= -\kappa\frac{\Delta t}{\Delta x}\frac{1}{\Delta x_{j-1/2}}, && j = 2, \ldots, N_j,
\end{aligned}$$

and $\boldsymbol{b}$ is a vector whose components are

$$\begin{aligned}
b_1 &= T_1^n + \Delta t\sigma_1^{n+1} + \kappa\frac{\Delta t}{\Delta x}\frac{1}{\Delta x_{1/2}}T_a^{n+1}, &&\\
b_j &= T_j^n + \Delta t\sigma_j^{n+1}, && j = 2, \ldots, N_j - 1,\\
b_{N_j} &= T_{N_j}^n + \Delta t\sigma_{N_j}^{n+1} + \kappa\frac{\Delta t}{\Delta x}\frac{1}{\Delta x_{N_j+1/2}}T_b^{n+1}. &&
\end{aligned}$$

Note that the BCs T_a and T_b are assumed known in the derivation of the FVM (cf. Section 2.2.3) and therefore appear in the right-hand side vector $\boldsymbol{b}$. Note also that the vector of free variables, $\boldsymbol{T}^{n+1}$, is only defined for interior grid nodes. Any source term designed to correct $\boldsymbol{T}^{n+1}$ can therefore only correct interior temperatures, not boundary temperatures. We fix this issue by increasing the dimensionality of the temperature vector such that it also describes boundary temperatures. We define the extended temperature vector as $\underline{\boldsymbol{T}} = [T_0, T_1, \ldots, T_{N_j}, T_{N_j+1}]$, where $T_0 = T(x_0)$ and $T_{N_j+1} = T(x_{N_j+1})$ are the boundary temperatures, which we now consider as free variables. Furthermore, let $\underline{\mathbb{A}}$ be

[1]The procedure for correcting BCs in higher-dimensional problems is analogous to the one described here.

an $N_j + 2 \times N_j + 2$-dimensional tridiagonal matrix with non-zero elements

$$\begin{aligned}
\underline{a}_{0,0} &= 1 \\
\underline{a}_{1,0} &= -\kappa \frac{\Delta t}{\Delta x} \frac{1}{\Delta x_{1/2}} \\
\underline{a}_{j,j} &= 1 + \kappa \frac{\Delta t}{\Delta x} \left(\frac{1}{\Delta x_{j+1/2}} + \frac{1}{\Delta x_{j-1/2}} \right), && j = 1, \dots, N_j, \\
\underline{a}_{j,j+1} &= -\kappa \frac{\Delta t}{\Delta x} \frac{1}{\Delta x_{j+1/2}}, && j = 1, \dots, N_j - 1, \\
\underline{a}_{j-1,j} &= -\kappa \frac{\Delta t}{\Delta x} \frac{1}{\Delta x_{j-1/2}}, && j = 2, \dots, N_j, \\
\underline{a}_{N_j,N_j+1} &= -\kappa \frac{\Delta t}{\Delta x} \frac{1}{\Delta x_{N_j+1/2}}, \\
\underline{a}_{N_j+1,N_j+1} &= 1,
\end{aligned}$$

and let $\underline{\boldsymbol{b}}$ be an $N_j + 2$-dimensional vector with components

$$\begin{aligned}
\underline{b}_0 &= T_a^{n+1} \\
\underline{b}_1 &= T_1^n + \Delta t \sigma_1^{n+1}, \\
\underline{b}_j &= T_j^n + \Delta t \sigma_j^{n+1}, && j = 2, \dots, N_j - 1, \\
\underline{b}_{N_j} &= T_{N_j}^n + \Delta t \sigma_{N_j}^{n+1} \\
\underline{b}_{N_j+1} &= T_b^{n+1}.
\end{aligned}$$

Note that we have used zero-indexing in the definitions of $\underline{\mathbb{A}}$ and $\underline{\boldsymbol{b}}$, such that any element in $\mathbb{A}$ or $\boldsymbol{b}$ can be found at the same index in $\underline{\mathbb{A}}$ or $\underline{\boldsymbol{b}}$ (with the exceptions of b_1 and b_{N_j}). With these definitions of $\underline{\mathbb{A}}$, $\underline{\boldsymbol{T}}$ and $\underline{\boldsymbol{b}}$,

$$\underline{\mathbb{A}}\, \underline{\boldsymbol{T}}^{n+1} = \underline{\boldsymbol{b}} \left(\underline{\boldsymbol{T}}^n \right) \tag{B.2}$$

is an equivalent formulation of Equation B.1, in the sense that the middle N_j components of $\underline{\boldsymbol{T}}^{n+1}$ are equal to the components of $\boldsymbol{T}^{n+1}$ if and only if the middle N_j components of $\underline{\boldsymbol{T}}^n$ are equal to the components of $\boldsymbol{T}^n$.

The FVM formulation (B.2) is never used in pure physics-based modelling because it is less computationally efficient than the formulation (B.1). However, from the perspective of CoSTA, the fact that the boundary temperatures are considered free variables in the formulation (B.2) is a major advantage, since we can then correct these boundary temperatures.

Suppose now that $\underline{\boldsymbol{T}}_{\text{ref}}$ denotes the true solution T_{ref} of a heat transfer problem evaluated at *all* grid nodes, including the grid nodes at the boundaries. Analogously to the regular corrective source term definition in Equation (3.14), we can then define an extended corrective source term as

$$\underline{\hat{\boldsymbol{\sigma}}}^{n+1} = \underline{\mathbb{A}}\, \underline{\boldsymbol{T}}_{\text{ref}}^{n+1} - \underline{\boldsymbol{b}} \left(\underline{\boldsymbol{T}}_{\text{ref}}^n \right). \tag{B.3}$$

We then get a modified system

$$\underline{\mathbb{A}}\, \underline{\boldsymbol{T}}_{\text{h}}^{n+1} = \underline{\boldsymbol{b}} \left(\underline{\boldsymbol{T}}_{\text{h}}^n \right) + \underline{\hat{\boldsymbol{\sigma}}}^{n+1}, \tag{B.4}$$

which is analogous to the modified system (3.15). The advantage of the definitions presented here is that the components $\underline{\hat{\sigma}}_0$ and $\underline{\hat{\sigma}}_{N_j+1}$ of $\underline{\hat{\boldsymbol{\sigma}}}$ allow us to correct the boundary temperatures if the BCs T_a and T_b prescribed by the original PBM are incorrect.

C. A Note on Conducting Grid Refinement Studies with Data-Driven or Hybrid Models

When grid refinement studies are to be performed for data-driven or hybrid models, it is vital to consider that these models require training before they can be deployed. Additionally, it is not guaranteed that any individual model can be used to make predictions for several discretizations unless specifically trained to do so. This stands in contrast to PBM, where the discretization can be altered without having to adjust the model in any other way. It also raises some fundamental questions regarding how a grid refinement study should be performed for models with data-driven components. We frame these questions in the context of DDM and HAM models based on DNNs, as that is the kind of models we have studied in detail in the present work. The questions are:

Q1 Are the DNN's architecture and training routine defined such that the DNN can be used with different discretizations?

Q2 If the answer to Q1 is yes: Should the grid refinement study use the same or different discretizations than those seen during training?

Q3 Should a unique set of hyperparameters be used for each discretization?

These are all open questions, and we do not claim to have perfect answers to them. However, they must be considered before any grid refinement study with DNN-based DDM or HAM models can be conducted. Below, we discuss our thoughts on these questions.

First, we observe that the answer to Q1 is specific to the DNN architecture and training routine used in the particular model that is studied. In the present work, the studied DDM and HAM models have utilized DNNs with a fully-connected architecture (cf. Section 4.1.4). For such an architecture, the number of parameters belonging to each neuron in the first hidden layer depends on the dimensionality of the input layer. For that reason, the DDM and HAM models we have considered all require input at a fixed resolution. It is therefore not straightforward to use them in a grid refinement study. Generally, one is then left with two options: 1) Train one model instance per resolution, with each model instance having input and output layers that match its resolution 2) Train one model instance whose input and output layers match the finest resolution, and upscale the coarse data to this resolution for use with the DNN.[1] In Section 4.5, we went with option 1), but we do not really have any theoretical reason for choosing one of these options over the other; they are both valid approaches worthy of greater consideration than could be given within the scope of the present work. However, it should be noted that option 1) may not be practically feasible for models which are computationally expensive to train.

[1]Upscaling can e.g. be performed using various interpolation techniques. If the DNN's neurons do not have biases, it is also possible to pad or dilute the input vectors with zeroes to increase their dimensionality.

It is worth noting that there do exist DNNs for which the answer to the first question is undoubtedly “yes”. So-called *convolutional layers* (see (Goodfellow et al., 2016, chapter 9) for an introduction) operate independently of their inputs’ dimensionality. Thus, any DDM or hybrid model based on a DNN with only convolutional layers will be *compatible* with multiple discretizations of the spatial domain. By compatible, we mean that these models *can* operate on data sampled using different discretizations. Whether or not their learning will generalize to different discretizations than those used to generate the data can only be determined through empiric investigations. Still, one would at minimum expect the model to work for any discretization used to define its training data. Whether or not it would generalize to other discretizations is an open question that is closely related to Q2 in the list above.

The answer to Q2 depends greatly on what it is we want to achieve with the grid refinement study. If we want the DNN to work for data sampled at varying resolutions, it could be a good idea to use different discretizations in the grid refinement study than during training. However, this would make the study much “harder” for the DNN, compared to using the same discretizations. Therefore, if we are not interested in this kind of generalization, it is probably best to use the same discretizations. Otherwise, we would just be diminishing the model’s accuracy for no good reason.

Finally, we consider Q3, which is clearly an important question for us, given that we opted to train individual model instances for each resolution in Section 4.5. Ideally, we would want each model instance to be perfectly tuned for the discretization used to generate the data it operates on. However, performing full parameter tuning processes for all discretizations considered would be prohibitly expensive in terms of computational cost. Furthermore, we do not have any heuristic for adjusting DNN hyperparameters to accommodate different discretizations either. For these reasons, we simply used the same hyperparameters for all discretizations. More specifically, we used the same hyperparameters as in all our other experiments: the hyperparameters of Table 4.4. Note that we did not make this choice because we have reason to believe it is the best choice, but rather because we could not find any better option. If grid refinement studies are going to be a staple analysis method for data-driven or hybrid models, more consideration should be given to questions Q1–Q3 than the scope of the present work permits.

D. Recovering an Unknown Wave Speed Using the CoSTA Framework

So far, we have only discussed how to recover unknown properties of the heat equation using DNN-generated corrective source terms. However, the main points of the discussion in Section 4.6 hold for other physical systems as well. This includes systems whose governing equation is mathematically equivalent to the heat equation (cf. Section 2.2), and systems with completely different governing equations. Here, we briefly illustrate how to recover unknown properties of systems governed by the wave equation.

The wave equation is an important equation in classical physics governing the propagation of both mechanical (e.g. sound or other vibrations) and electromagnetic (e.g. light) waves (Griffiths, 2017, chapter 9). For a general variable u, it reads

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u, \tag{D.1}$$

where c is the wave speed. Suppose now that the true wave speed c is approximated by $\tilde{c} = c - \epsilon_c$. The true wave equation can then be written as

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u = (\tilde{c} + \epsilon_c)^2 \nabla^2 u = \tilde{c}^2 \nabla^2 u + (2\tilde{c}\epsilon_c + \epsilon_c^2)\nabla^2 u, \tag{D.2}$$

while the wave equation of the model based on the approximate wave speed $\tilde{c}$ is

$$\frac{\partial^2 \tilde{u}}{\partial t^2} = \tilde{c}^2 \nabla^2 \tilde{u}, \tag{D.3}$$

where $\tilde{u}$ is the model's approximation of the true solution u. Comparing the latter equation to Equation (3.3), we have $\widetilde{\mathcal{N}}_\Omega = \partial^2/\partial t^2 - \tilde{c}^2\nabla^2$ and $\tilde{f} = 0$. Equation (D.2) can then be rewritten as

$$\frac{\partial^2 u}{\partial t^2} - \tilde{c}^2 \nabla^2 u = \widetilde{\mathcal{N}}_\Omega u = (2\tilde{c}\epsilon_c + \epsilon_c^2)\nabla^2 u, \tag{D.4}$$

Upon comparison with Equation (3.5), we can then conclude that the corrective source term is

$$\hat{\sigma} = (2\tilde{c}\epsilon_c + \epsilon_c^2)\nabla^2 u. \tag{D.5}$$

If the incorrect modelling of c is the model's primary source of error, we can make the replacements $\epsilon_c \to \hat{\epsilon}_c$ and $u \to \hat{\tilde{u}}$ in the equation above to obtain

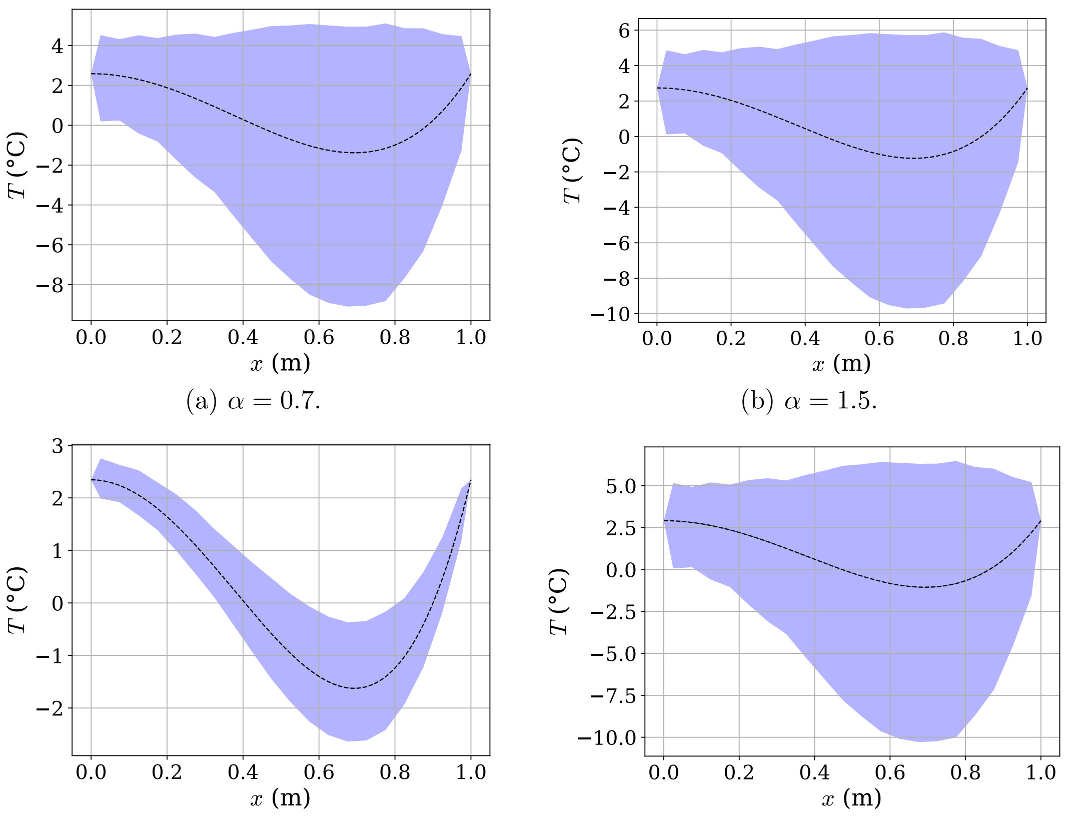
$$\hat{\sigma}_{\text{NN}} \approx (2\tilde{c}\hat{\epsilon}_c + \hat{\epsilon}_c^2)\nabla^2 \hat{\tilde{u}}. \tag{D.6}$$

Here, $\hat{\sigma}_{\text{NN}}$ is the predicted corrective source term, $\hat{\tilde{u}}$ is the corrected model's approximation of u, and $\hat{\epsilon}_c$ is the predicted error in the modelling of c. After solving the above for $\hat{\epsilon}_c$, we can (approximately) recover the true wave speed as $c \approx \hat{c} = \tilde{c} + \hat{\epsilon}_c$ from $\hat{\sigma}_{\text{NN}}$.

We observe that Equation (D.6) generally has two solutions for $\hat{\epsilon}_c$. Some physical intuition may therefore be required for choosing which solution to use when defining the corrected wave speed $\hat{c} = \tilde{c} + \hat{\epsilon}_c$. However, in scenarios where only one solution yields a physically viable $\hat{c}$, the choice is obvious.

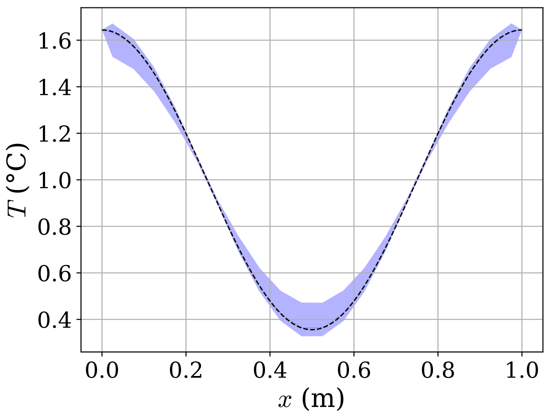
E. Additional Results Regarding Predictive Uncertainty

In Section 4.4, we provided some visualizations of the inherent uncertainty of the DDM and CoSTA-based HAM models used in the experiments described in Section 4.2.2. These visualizations (i.e. Figures 4.43 and 4.44) contain coloured areas whose sizes correspond to standard deviations of the models' predictions for Solutions P1 and P3. The analogous visualization for Solutions P2 and P4 are shown in Figures E.1 and E.2. As claimed in Section 4.4, the areas supposed to visualize the uncertainty of the HAM model are so small that they are not visible in these figures. For this reason, they are not really successful as visualizations, and that is why they are included here rather than in the main text.

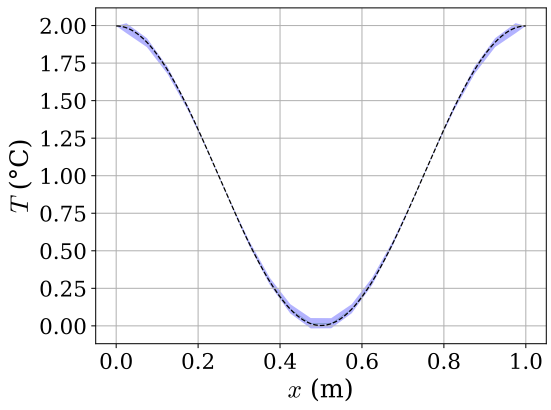


(a) $\alpha = 0.7$. (b) $\alpha = 1.5$. (c) $\alpha = -0.5$. (d) $\alpha = 2.5$.

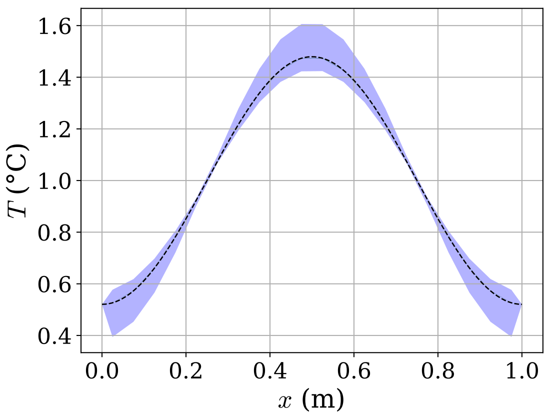
Figure E.1.: Solution P2: Visualization of the inherent uncertainty of DDM and HAM models, where a large coloured area corresponds to a large model uncertainty. Blue area corresponds to uncertainty of DDM, and green area (not visible) corresponds to uncertainty of HAM. The dashed line represents the exact solution P2.



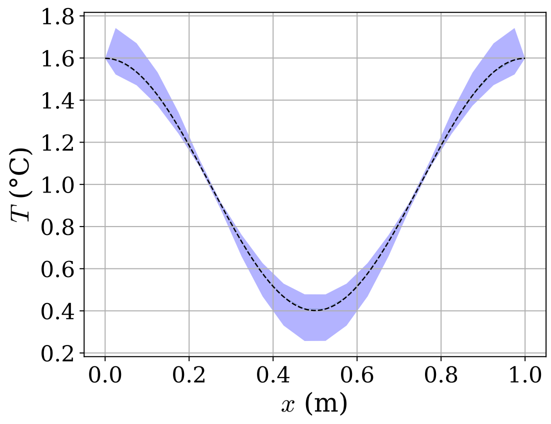
(a) $\alpha = 0.7$.



(b) $\alpha = 1.5$.



(c) $\alpha = -0.5$.



(d) $\alpha = 2.5$.

Figure E.2.: Solution P4: Visualization of the inherent uncertainty of DDM and HAM models, where a large coloured area corresponds to a large model uncertainty. Blue area corresponds to uncertainty of DDM, and green area (not visible) corresponds to uncertainty of HAM. The dashed line represents the exact solution P4.

F. Estimating the Relative Impact of Modelling Error and Discretization Error in 1D Experiments

For the experiments considered in Section 4.2.2, the PBM is affected by both discretization error and modelling error. Here, we aim to determine the relative impact of these two error sources. (For completeness, similar analyses for the experiments considered in Sections 4.2.3, 4.3.1 and 4.3.2 would have been desirable, but such analyses were not conducted due to time constraints.)

For a given manufactured solution T_{ref}, we compute the modelling error $\mathcal{E}_{\text{mod}}$ of a PBM with $P = 0$ as

$$\mathcal{E}_{\text{mod}} = \|\boldsymbol{T}_{\text{p}}(P = 0, t_{\text{end}}) - \boldsymbol{T}_{\text{ref}}(t_{\text{end}})\|_2 / \|\boldsymbol{T}_{\text{ref}}(t_{\text{end}})\|_2, \tag{F.1}$$

where $\boldsymbol{T}_{\text{p}}$ and $\boldsymbol{T}_{\text{ref}}$ must be defined on a very fine grid to reduce the influence of the discretization error. In our calculations of $\mathcal{E}_{\text{mod}}$, we use $N_j = 1000$ and $N_t = 50000$, which is a very fine discretization compared to the discretization with $N_j = 20$ and $N_t = 5000$ used in Section 4.2.2.

We compute the discretization error $\mathcal{E}_{\text{discr}}$ as

$$\mathcal{E}_{\text{discr}} = \|\boldsymbol{T}_{\text{p}}(P, t_{\text{end}}) - \boldsymbol{T}_{\text{ref}}(t_{\text{end}})\|_2 / \|\boldsymbol{T}_{\text{ref}}(t_{\text{end}})\|_2, \tag{F.2}$$

where we use same discretization as in Section 4.2.2, and where the true value of P is used in the PBM.

The modelling and discretization errors of Solutions P1, P2, P3 and P4 (cf. Table 4.1) are compared in Table F.1. We observe that the modelling error is dominant for all four solutions.

Table F.1.: Comparison of modelling error and discretization error for the solutions considered in the experiments of Section 4.2.2.

Solution	$\mathcal{E}_{\text{mod}}$	$\mathcal{E}_{\text{discr}}$	$\frac{\mathcal{E}_{\text{mod}}}{\mathcal{E}_{\text{mod}}+\mathcal{E}_{\text{discr}}}$	$\frac{\mathcal{E}_{\text{discr}}}{\mathcal{E}_{\text{mod}}+\mathcal{E}_{\text{discr}}}$
P1	3.48e-2	6.41e-3	0.8445	0.1555
P2	1.76e0	1.19e-2	0.9933	0.0067
P3	6.44e-2	4.61e-4	0.9929	0.0071
P4	6.20e-1	1.26e-2	0.9800	0.0200

G. Technical Details of the 2D Implicit Euler FVM

In the main text, we did not define the temperature vector $\boldsymbol{T}$, the coefficient matrix $\mathbb{A}$ or the right-hand-side vector $\boldsymbol{b}$ of the 2D Implicit Euler FVM on matrix form (Equation (2.21)). Due to space concerns, their definitions are instead given here. From a geometric perspective, the matrix $\mathbb{A}$ in Equation (2.21) can be viewed as a block-tridiagonal matrix. Then, both $\mathbb{A}$ and its main-diagonal blocks become tridiagonal matrices similar to $\mathbb{A}$ in Equation (2.18). Moreover, the sub- and super-diagonal blocks are then diagonal matrices. From this same perspective, both $\boldsymbol{b}$ in Equation (2.21) and its constituent blocks take the form of $\boldsymbol{b}$ in Equation (2.18). However, these block-formulations are not particularly useful for implementing the 2D Implicit Euler FVM. We therefore define $\mathbb{A}$ and $\boldsymbol{b}$ algorithmically, as one would do when writing a computer implementation of the FVM. $\mathbb{A}$ is defined in Algorithm 3, while $\boldsymbol{b}$ is defined in Algorithm 4. Note that we use % to denote the modulo operator, and that we write "//" to denote line comments.

Algorithm 3: Algorithm for defining $\mathbb{A}$ in Equation (2.21).

```
Initialize $\mathbb{A}$ as an $N_jN_i \times N_jN_i$ matrix of zeros.
for $j = 1, 2, \ldots, N_jN_i$ do
    for $i = 1, 2, \ldots, N_jN_i$ do
        // Define main diagonal of $\mathbb{A}$
        if $i = j$ then
            $a_{j,i} \leftarrow 1 + \frac{2\kappa\Delta t}{\Delta x^2} + \frac{2\kappa\Delta t}{\Delta y^2}$
            if $i \leq N_j$ or $i > N_j(N_i - 1)$ then
                $a_{j,i} \leftarrow a_{j,i} + \frac{\kappa\Delta t}{\Delta y^2}$
            end
            if $j\%N_j = 0$ or $j\%N_j = 1$ then
                $a_{j,i} \leftarrow a_{j,i} + \frac{\kappa\Delta t}{\Delta x^2}$
            end
        end
        // Define 1st super-diagonal and sub-diagonal of $\mathbb{A}$
        if $i = j + 1$ or $i = j - 1$ then
            if $\max\{i, j\}\%N_j \neq 1$ then
                $a_{j,i} \leftarrow -\frac{\kappa\Delta t}{\Delta x^2}$
            end
        end
        // Define $N_j$th super-diagonal and sub-diagonal of $\mathbb{A}$
        if $i = j + N_j$ or $i = j - N_j$ then
            $a_{j,i} \leftarrow -\frac{\kappa\Delta t}{\Delta y^2}$
        end
    end
end
```

Algorithm 4: Algorithm for defining $\boldsymbol{b}$ in Equation (2.21).

```
Initialize $\boldsymbol{b}$ as an $N_jN_i$-component vector of zeros.
for $j = 1, 2, \ldots, N_jN_i$ do
    $b_j \leftarrow \boldsymbol{T}_j^{n+1} + \Delta t \boldsymbol{\sigma}_j^{n+1}$
    // Account for left BC.
    if $j\%N_j = 1$ then
        $b_j \leftarrow b_j + \frac{2\kappa\Delta t}{\Delta x^2} T_a^{n+1}$
    end
    // Account for right BC.
    if $j\%N_j = 0$ then
        $b_j \leftarrow b_j + \frac{2\kappa\Delta t}{\Delta x^2} T_b^{n+1}$
    end
    // Account for bottom BC.
    if $j \leq N_j$ then
        $b_j \leftarrow b_j + \frac{2\kappa\Delta t}{\Delta y^2} T_c^{n+1}$
    end
    // Account for top BC.
    if $j > N_j(N_i + 1)$ then
        $b_j \leftarrow b_j + \frac{2\kappa\Delta t}{\Delta y^2} T_d^{n+1}$
    end
end
```

Finally, we must define the temperature vector $\boldsymbol{T}$. Generally, the temperature $T_{j,i}$ at the grid node (x_j, y_i) can be found at index $i \cdot N_j + j$ in $\boldsymbol{T}$.

H. Tables in Landscape Format

Due to their large width, a very small font size was required for Tables 4.1 and 4.2 to fit the page. When reading on screen, this is not an issue, since it is then possible to zoom in to make the writing larger. However, it may pose a significant problem when reading in print. To the benefit of readers with a printed copy of this thesis, duplicates of Tables 4.1 and 4.2 in landscape format are provided below. The use of landscape mode allows for a larger font size, which improves readability on paper. Note that tables in landscape mode are difficult to read on fixed screens, and generally inhibit reading flow regardless of medium. These are the reasons why the tables were presented using the standard portrait format in the main text.

Table H.1.: Duplicate of Table 4.1, but now in landscape format for improved readability in print.

Label	$T_{\text{ref}}(x,t;\alpha)$	$P(x,t;\alpha)$	$k(x,t;\alpha)$
d1	$\alpha\left(t+\frac{1}{2}x^2\right)$	0	1
P1	$t+\frac{1}{2}\alpha x^2$	$1-\alpha$	1
P2	$\sqrt{t+\alpha+1}+10x^2(x-1)(x+2)$	$\frac{1}{2\sqrt{t+\alpha+1}}-120x^2-60x+40$	1
P3	$2+\alpha(x-1)\tanh\left(\frac{x}{t+0.1}\right)$	$\frac{\alpha}{(t+0.1)^2}\Big(x(1-x)+2\Big((x-1)\tanh\left(\frac{x}{t+0.1}\right)-t-0.1\Big)\Big)\operatorname{sech}^2\left(\frac{x}{t+0.1}\right)$	1
P4	$1+\sin(2\pi t+\alpha)\cos(2\pi x)$	$2\pi\left(\cos(2\pi t+\alpha)+2\pi\sin(2\pi t+\alpha)\right)\cos(2\pi x)$	1
k1	$t+\alpha x$	$1-\alpha$	$1+x$
k2	$5-\frac{\alpha x}{1+t}$	$\frac{\alpha(x-\alpha)}{(1+t)^2}$	$5-\frac{\alpha x}{1+t}$
k3	$e^{-t}\cdot\begin{cases}\alpha+2x, & x\le 0.5\\ \alpha+0.75+0.5x, & x>0.5\end{cases}$	$-e^{-t}\cdot\begin{cases}\alpha+2x, & x\le 0.5\\ \alpha+0.75+0.5x, & x>0.5\end{cases}$	$\begin{cases}0.5, & x\le 0.5\\ 2, & x>0.5\end{cases}$
k4	$4x^3-4x^2+\alpha(t+1)$	$\alpha-36x^2-8x+8$	$1+x$
A	$\sqrt{t+\alpha+1}+7x^2(x-1)(x+2)$	$\frac{1}{2\sqrt{t+\alpha+1}}-84x^2-42x+28$	1
B	$-\frac{x^3(x-\alpha)}{t+0.1}$	$\frac{x^4-\alpha x^3}{(t+0.1)^2}+\frac{12x^2-6\alpha x}{t+0.1}$	1

Table H.2.: Duplicate of Table 4.2, but now in landscape format for improved readability in print.

Label	$T(x,y,t;\alpha)$	$P(x,y,t;\alpha)$	$k(x,y,t;\alpha)$
2P1	$t+0.5\alpha(x^2+y^2)+x$	$(1-2\alpha)$	1
2P2	$1+\sin(2\pi t+\alpha)\cos(2\pi x)\cos(2\pi y)$	$2\pi\cos(2\pi x)\cos(2\pi y)\left(\cos(2\pi t+\alpha)+4\pi\sin(2\pi t+\alpha)\right)$	1
2k1	$t+\alpha x+y^* * 2$	$-(1+\alpha+2x+4y)$	$1+x+y$
2k2	$\alpha+(t+1)\cos(2\pi x)\cos 4\pi y$	$\cos(2\pi x)\cos(4\pi y)\left(1+40\pi^2(t+1)\left(1+\sin(1\pi x)\sin(4\pi y)\right)\right)$	$2+\sin(2\pi x)\sin(4\pi y)$

Sindre Stenen Blakseth

Introducing the Corrective Source Term Approach

NTNU
Norwegian University of
Science and Technology