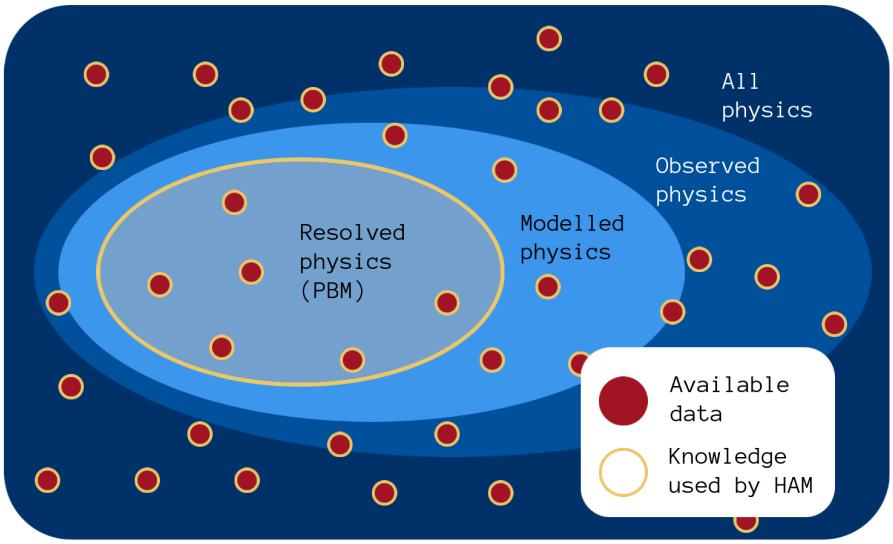
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Sondre Sørbø

Corrective Source Term Approach for Improving Physics Based Models

Master's thesis in Applied Physics and Mathematics
Supervisor: Trond Kvamsdal and Adil Rasheed
June 2022



NTNU
Norwegian University of Science and Technology

Sondre Sørbø

Corrective Source Term Approach for Improving Physics Based Models

Master's thesis in Applied Physics and Mathematics
Supervisor: Trond Kvamsdal and Adil Rasheed
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU

Kunnskap for en bedre verden

Abstract

Reliable predictive modelling techniques are paramount to innumerable industry objectives. With the ever emerging big data revolution, traditional methods based on physical descriptions are being challenged by the newer machine learning paradigm. Still, many industries are reluctant to trusting black box models, with good reasons. Although these models in numerous occasions offer more accurate predictions at a lower computational cost in the online phase of the model, they come with significant disadvantages in terms of trustworthiness and generalizability.

Hybrid analysis and modelling aim to combine ideas from physics based and data driven modelling into more robust hybrid models. In this thesis we conduct a series of experiments on modelling heat and elasticity using one of these hybrid methods, the corrective source term approach (CoSTA), to correct for various modelling errors stemming from ignored dimensions, PDE linearization, unknown source terms, or just from discretization of the equation. We show that CoSTA produce significantly more accurate results than its counterpart physics based or data driven models, also when generalizing to situations qualitatively different from the training data. Although CoSTA models has random variations, we argue that the predictions are more than sufficiently consistent to outclass both the physics based and the data driven models.

Samandrag

Pålitelege prediksjonsmodelleringsteknikkar er kritisk for eit utal av oppgåver i industrien. Med ein stadig pågåande teknologisk revolusjon innen store datamengder, vert tradisjonelle metodar basert på fysiske beskrivingar utfordra av det nyare maskinlæringsparadigmet. Samstundes nøler mange industriar med å lita på modellar basert på ukjende berekningar, med god grunn. Sjølv om desse modellane i atskillege tilfelle gjev meir nøyaktige prediksjonar, og krev mindre utrekning når modellane fyrst har vorte trena, så kjem dei med betydelege ulemper når det gjeld pålitelegheit og generaliseringsevner.

Hybrid analyse og modellering siktar på å kombinera idear frå fysikkbaserte og datadrivne modellar til meir robuste hybride modellar. I denne oppgåva utfører me ei rekkje eksperiment der me modellerer varme og elastisitet, ved å nytta ein av desse hybride metodane, CoSTA (frå engelsk "corrective source term approach"), for å korrigera for modelleringsfeil som stammar frå ignorerte dimensjonar, linearisering, ukjende kjeldeledd eller berre frå diskretisering av likningane. Me viser at CoSTA gir mykkje meir nøyaktige resultat enn tilsvarande fysikkbaserte eller datadrivne modeller, òg når ein generaliserer til situasjonar som er kvalitativt ulike frå treningsdataene. Sjølv om CoSTA-modellar har vilkårlege variasjonar, argumenterer me for at prediksjonane er meir enn konsistente nok til å utklassa både dei fysikkbaserte og dei datadrivne modellane.

Preface

This master thesis is written in the spring of 2022, and concludes my degree in physics and mathematics at NTNU. All experiments and results in this thesis were made during this semester. Still, the project is in some ways a continuation of my specialization project [1] the preceding fall, which used most of the same methods. In particular, an earlier version parts of Chapter 2 was presented in the report of that project.

I would like to thank my supervisors, Professor Trond Kvamsdal and Professor Adil Rasheed, for their time, guidance, and support throughout both this project, and the mentioned specialization project. They have both been essential contributors to the success of this project. Adil had the ideas for the experiments, and has introduced me to the concepts of hybrid analysis and modelling. His optimism and ambition at all stages of the project has been a great encouragement. Trond has taught me most of what I know about the finite element method, which is a big part of this project, both in a course in 2020 and during his guidance. He has also helped me understand the partial differential equations used in this thesis, and how to model them.

Finally, I thank my lovely wife, and my friends and family, who certainly have supported me throughout this project and the preceding years of study, but more importantly, have distracted me and helped me think about other matters once in a while.

Sondre Sørbø
Trondheim, June 2022

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

CNN	Convolutional neural network	2
CoSTA	Corrective source term approach	2
DDM	Data driven modelling	1
DNN	Deep neural network	9
FEM	Finite element method	4
HAM	Hybrid analysis and modelling	2
ML	Machine learning	1
MSE	Mean square error	16
PBM	Physics based modelling	1
PGDA	Physics-guided design of architecture	2
PGI	Physics-guided initialisation	2
PGLF	Physics-guided loss function	2
ReLU	Rectified linear unit	15
RRMSE	Relative root mean square error	16

Symbols

α	Parameter the manufactured solutions depend on	13
$\bar{\boldsymbol{\epsilon}}$	Linear operator defined in Equation (2.3.4)	7
$\bar{\boldsymbol{\sigma}}$	Vector form of the Cauchy stress tensor	7
$\boldsymbol{u}$	Vector field PDE solution	6
ϵ	Strain tensor	6
λ	Borel measure of spacial domain	8
$\mathcal{A}$	Set of values of α	13
ν	Poisson's ratio	6
Ω	Spacial domain	5

Chapter 1
Introduction

Rarely do we have a complete understanding of any phenomena of interest. Moreover, due to numerous assumptions needed to make solutions computationally tractable, predictions are erroneous. In such a situation, a model that can address the problems of incomplete models and drastic assumptions will be highly welcome. This thesis demonstrates how the Corrective Source Term Approach can address these problems.

1.1 Motivation and Background

Predictive modelling and simulation has for a long time mainly been done using physics based modelling (PBM). With the rise of machine learning (ML) in the last decades, data driven modelling (DDM)[1] has shown its ability to outperform PBM in many situations [2, 3, 4, 5], but it comes with its own disadvantages, hurting the overall usefulness of these methods. In [6] the authors describe the ideal model in the context of digital twins as generalizable, trustworthy, self-evolving, computationally efficient and accurate. Although the importance of the various properties of a model depends on its intended use, these properties should be desirable in any situation. Likewise, unwanted properties are the opposite of these, along with requirements like large amounts of data.

PBMs, when based on the correct physics, are very accurate and generalizable but are usually computationally demanding and do not adapt to new information automatically. DDMs, on the other hand, after training, are very efficient and possibly very accurate but not generalizable. The accuracy of both methods is very dependent on the knowledge used to build the model. PBMs require accurate knowledge of the physical phenomenons, as well as mathematical methods for solving the physical equations. We can come quite close to the perfect model described above with accurate physics and analytical or good numerical solutions. However, if the physical knowledge is inaccurate, or we lack methods for efficient computation, this heavily affects the model's performance. While DDMs do not need any physical knowledge, they require data, usually in relatively large amounts, and the model's accuracy is highly dependent on how this data represents the scenarios for which the model is to be used. With good data, DDM can deliver highly accurate results for only a fraction of the computational cost of similar results from a PBM, if there even is a PBM available. However, DDM perform poorly on situations that differ from the training data. Another problem with DDM is the randomness and unexplainability of the results. Each initialization of a DDM is unique, and its results will vary slightly from the others for the same problem. Moreover, because of the black-box nature of the DDMs, there is no guarantee for the models predictions to be accurate in all scenarios. This last problem with DDM has prevented them from being utilized in many high-stake applications where a single inaccurate result has a significant impact.

The increasing usage of digital twins, which we observe across the industries, brings with it a great demand for models capable of accurately modelling processes in real time [7]. But with their apparent deficiencies, PBM and DDM lack the abilities needed to fully reap

[1]The abbreviation DDM will be used both as the concept of data driven *modelling*, and a specific data driven *model*. It should be clear from the context which one we mean. PBM is also used in this way.

the benefits of the digital twins. To counter these problems, hybrid analysis and modelling (HAM) has been and is emerging as a new eclectic paradigm that combines techniques from both PBM and DDM. The aim is to combine the strengths of both paradigms to create better models, that obtain the desirable properties described above.

1.1.1 Literature Review

Hybrid Analysis and Modelling

How to best combine physics based and data driven methods in a beneficial manner highly depends on the situation, and several different successful HAM techniques have shown promising results. Be aware of inconsistent nomenclature when reading articles by different authors. [8] provides a solid list of contributions to this field of research, and systematically organise them based on both the methods of integrating physical knowledge in machine learning, and the type of problem it is used to solve. This section is based on that survey. The objectives when using HAM can be to improve properties (accuracy, computational efficiency e.g.) of PBMs, parts of PBMs or reduced order models, or for downscaling. Other objectives are to generate data for usage in subsequent data-demanding applications, and to use available (possibly synthetic) data to infer properties of the governing equation or the PBM, like inferring physical parameters, discovering governing equations, or quantifying uncertainty of a PBM.

[8] also divides the *methods* into four groups. Physics-guided loss function (PGLF), where a penalty for nonphysical output is added to the loss function, is a straight-forward way of enforcing a DDM to produce output mostly consistent with physical knowledge. This method has been used for all the objectives listed above [9, 10, 11, 12, 13, 14, 15, 16, 17]. In physics-guided design of architecture (PGDA), the physical knowledge is incorporated into the architecture of the DDM in some way. This may involve having intermediate nodes in a DNN take values of parameters known to be important to the system, or enforcing invariances like translational or rotational symmetries using custom convolutional kernels in a convolutional neural network (CNN). PGDA methods are also used for all objectives listed above [18, 19, 20, 21, 22, 23, 24, 25, 26]. Physics-guided initialisation (PGI) methods use physical knowledge when generating the initial values of the parameters in a DDM, before training. Physical data (often synthetic) can be used for pre-training a DDM (or parts of it) by using transfer learning. PGI has been used for improving physical models [27, 28, 29, 30], and for uncertainty quantification [31, 32]. Common for all these methods is the concept of utilising the physical knowledge to augment a DDM. The alternative is to base the method on either a PBM or the combination of a PBM and a DDM. Such techniques can be, among other, to use a DDM to predict the error of a PBM [33, 34, 35, 36], using the output of a PBM as input in a DDM [37], replacing components of PBMs with DDMs [38, 39, 40], or using an ensemble method consisting of both PBM and DDM [41, 42, 43].

The Corrective Source Term Approach

The corrective source term approach (CoSTA), originally introduced in [44], combines the strengths of PBM and DDM by using the DDM as a correction for parts of the physical processes not covered by the PBM, through applying it as a source term correction before solving with the PBM. This is similar to the error predicting method mentioned above, but the DDM is applied before solving the PBM. The method also uses the output of uncorrected the PBM as input. [45] gives a quick and precise introduction and theoretical justification of the method, along with impressive results in several one-dimensional heat modelling problems. [44] also

includes results for two-dimensional problems, and concludes that CoSTA provides a substantial increase in accuracy over both PBM and DDM, and is more generalizable than DNN. They also argue that by interpreting the corrective source term, automated sanity checks can make sure the method produces trustworthy results, unlike most DDMs.

1.2 Objectives and Research Questions

The previous works about CoSTA has argued that CoSTA can be applied to any deterministic PBM. Being a partially black box method without theoretical results describing the quality of its performance, its trustworthiness relies on empirical results. Yet, with the method still being in its infancy, the results provided thus far are relatively few, and cover a narrow selection of problems relative to its claimed area of applicability. The purpose of this thesis is to provide more results on the performance of CoSTA, and expand the set of problems it has been used to solve. To this end, we conduct a series of experiments, that differ from previous experiments by

- including new types of modelling errors to the PBMs, stemming from linearization and ignoring dimensions of a PDE,
- modelling elasticity as well as heat conduction,
- using the finite element method, as opposed to the of finite difference method, to form the PBM.

In each experiment we test these properties of CoSTA, along with its corresponding model parts PBM and DDM:

- Accuracy - How far is the prediction from the correct solution?
- Generalizability - To what extent do the models performance depend on the similarity between the training data and predictions?
- Consistency - How much do the predictions from unique CoSTA models, trained on the same data, vary?

Through answering these questions in each experiment, we ultimately aim to answer the main question: Does the previously published results from CoSTA generalize to these situations?

1.3 Outline of Report

This thesis includes an introduction to the relevant theory in Chapter 2. A person with a mathematical background should be able to understand the concepts quite well, although for using the methods seen in this thesis for one's own experiments it might be necessary to check out some of the sources cited in that chapter, especially for someone without experience with using neural networks.

Chapter 3 presents setup of how the experiments in the thesis were performed, first in general, then the details of each experiments are presented. This chapter should make the reader able to reproduce results similar to the ones presented in this thesis. Chapter 4 presents the results of the experiments and a discussion of these. Section 4.1 presents dimensional reduction of the heat equation, followed by linearization of the heat equation in Section 4.2. In Section 4.3 we model the elasticity equations with and without source term. The same equation is modelled with reduced dimensionality in Section 4.4 and linearization in Chapter 4.5. Finally, we summarize and discuss all the collected results in Section 4.6 before presenting our conclusion and suggestions for future work in Chapter 5.

Chapter 2
Theory

In this chapter the theory on which this thesis is based on, is presented. Please note that an earlier version of most of this chapter (not including the sections about elasticity) was already presented in the specialisation project [1].

2.1 Physics Based Modelling

Physics based modelling involves careful observation of a physical phenomenon of interest (heat diffusion and elasticity in the current work), development of its partial understanding, expression of the understanding in the form of mathematical equations (based on e.g. the law of conservation of energy) and ultimately solution of these equations. Due to the partial understanding and numerous assumptions along the steps from observation to solution of the equations, a large portion of the important governing physics gets ignored. Due to high computational costs, high fidelity simulators with minimal assumptions has so far been limited to the offline design phase only. Despite this major drawback, what makes these models attractive are sound foundations from first principles, interpretability, generalizability and existence of robust theories for the analysis of stability and uncertainty. However, most of these models are generally computationally expensive, do not adapt to new scenarios automatically and can be susceptible to numerical instabilities. The PBMs used in this thesis are discretizations of the modelled PDEs using the finite element method (FEM) along the spatial dimension, and the backward Euler method along the temporal dimension. For an introduction to FEM, see [46] or [47]. In addition to the inevitable discretization error, we will in most experiments in this thesis have some intentional modelling error, from PDE linearization, ignored dimensions or ignored source terms. The purpose of this is to test CoSTA on realistic erroneous models. In real world applications, such modelling errors may be intentional (or rather a known price for an intentional simplification) in order to speed up calculations. But may also stem from unknown physics, or lack of information about the system necessary to take advantage of more complex physical models. In this thesis, we choose the heat equation and the elasticity equation to demonstrate the effectiveness of the CoSTA method.

2.2 Heat Diffusion Modelling

In this section we introduce the heat equation, the equation that has been modelled using CoSTA in [44] and [45]. We have chosen to continue the work on this problem, as in addition to enabling easy validation of the implementation by comparing to the results in [44, 45], heat modelling is highly relevant for many modelling situations, for several reasons. Firstly, temperature measurements are being increasingly used for health / condition monitoring on a large variety of applications. [48] demonstrated how body temperature can be used for remote health monitoring while [49] used temperature for real-time bearing condition monitoring. Secondly, many quantities that can not be directly measured in a process can be indirectly inferred from the measured temperature, and thirdly, non-intrusive, cost-effective methods exist to make temperature measurements in real-time (eg. [50]).

Governing Equation

The heat equation with constant unit thermal conductivity can be written as

$$\frac{\partial u}{\partial t} = \Delta u + f, \tag{2.2.1}$$

where $u = u(t, x)$ is the temperature and $f = f(t, x)$ is a heat source term, and $\Delta = \nabla^2$ is the Laplace operator. The equation states that heat will flow from warm to cold areas at a rate proportional to the heat gradient, while additional heat may be introduced or extracted using the source term. For a more thorough introduction to the heat equation, see for example [51]. We will consider the equation on a bounded spatial domain $x \in \Omega$, for a time interval $t \in [0, T]$. In order for the equation to yield a unique solution, we need an initial condition and a boundary condition,

$$u(0, x) = u_0(x) \quad \forall x \in \Omega \tag{2.2.2}$$

$$L_{\partial\Omega} u(t, x) = g(t, x) \quad \forall x \in \partial\Omega, t \in [0, T], \tag{2.2.3}$$

for some appropriate[1] differential operator $L_{\partial\Omega}$. In this thesis, we will use Dirichlet boundary conditions, i.e. $L_{\partial\Omega} = I$, where I is the identity operator.

2.2.1 Discretization with Finite Elements

We will now derive the discretized heat equation. We start by discretizing the time dimension, and then the spacial dimension(s).

Let k be some constant time step length, and define the time steps $t_i = ik$ for $i \in \{0, 1, ...\}$. Let $u_i(\cdot) \approx u(t_i, \cdot)$ be the numeric approximation of the solution at time t_i. We approximate the time derivative by $\frac{\partial u}{\partial t}(t_i) \approx \frac{u_i - u_{i-1}}{k}$ and get the semi-discretized heat equation

$$u_i - k\Delta u_i = u_{i-1} + f(t_i, \cdot). \tag{2.2.4}$$

We require the functions u_i to have integrable first derivatives, and consider the solution space $X = H^1(\Omega)$. Multiplying with a test function $v \in X_0 = \{v \in X : v|_{\partial\Omega} = 0\}$ and integrating over the spacial domain Ω, we get

$$\int_\Omega (u_i v - k_i v \Delta u) dx = \int_\Omega v(u_{i-1} + f(t_i, \cdot)) dx, \tag{2.2.5}$$

or equivalently, since $v|_{\partial\Omega} = 0$,

$$\int_\Omega (u_i v + k_i \nabla v \nabla u) dx = \int_\Omega v(u_{i-1} + f(t_i, \cdot)) dx. \tag{2.2.6}$$

We now consider the above equation with the functions in a finite dimensional subspace $X^h \subset X$ spanned by the basis of functions $\{\phi_j\}_j$. This means we have $u_i = \sum_j u_{i,j}\phi_j$, $v = \sum_j v_j \phi_j$, and $f(t_i, x) = \sum_j f_{i,j}\phi_j(x)$, for some finite sets of scalars $\{u_{i,j}\}_j, \{v_j\}_j, \{f_{i,j}\}_j$. Let $U^{(i)}$ and V be vectors of the two former sets. We can then rewrite the above equation into

$$V^T(M + kA)U^{(i)} = V^T M U^{(i-1)} + V^T F^{(i)}, \tag{2.2.7}$$

where M is the matrix with elements $m_{j,l} = \int_\Omega \phi_j \phi_l dx$, A the matrix with elements $a_{j,l} = \int_\Omega \nabla\phi_j \nabla\phi_l dx$ and $F^{(i)}$ the vector with elements $\int_\Omega \phi_j f(t_i, \cdot) dx$. We use a lifting function (see [46]) to handle the boundary conditions, and end up with the discretized equations

[1] There has to be Dirichlet condition on at least some part of $\partial\Omega$, or constant terms can be added to the solution without violating the heat diffusion equation, thus making the solution non-unique.

$$(M + kA)U^{(i)} = MU^{(i-1)} + F^{(i)}. \tag{2.2.8}$$

This system of linear equations can be solved by any appropriate solver, preferably one for sparse matrices if the basis allows it. Starting from U_0, the known initial condition u_0 projected onto X^h, this enables us to calculate an approximation of $u(x,t)$ step by step. The results in this thesis are found using piecewise linear Lagrange elements [52] on a equidistant grid. As long as the grid is fine enough, and the equation, the boundary and initial conditions and the source term is known, this gives a very accurate solution. However, we will see that when we introduce modelling errors, the solution might be quite far from the correct one.

2.3 Elasticity Modelling

The linear elasticity equations is a set of differential equations describing the continuous deformation of solid objects when subject to given forces. It is used extensively in civil and mechanical engineering. This linearized model is a good approximation when the deformation is small relative to the object size, and the stress is linearly dependent on the stain, for forces small enough to not permanently deform the objects. A solid introduction to this equation and the theory from which it is derived, can be found in [53].

We will consider both the two-dimensional and three-dimensional versions, so we start by describing a general n-dimensional system. We will in this thesis assume unit mass density and Young's modulus in all cases[2]. We will also assume the material is isotropic, meaning its properties are independent of orientation (unlike e.g. wood which is stronger along the grains), and that Poisson's ratio is $\nu = 0.25$, a value close to those of many typical solids. Let $\boldsymbol{u}$ be the displacement vector field. Its elements describe a points position relative to where it would be without the applied forces. Furthermore, let ϵ be the strain tensor, a $n \times n$ matrix. Its elements describe the relative stretching of an infinitesimal element of the object. Finally, the Cauchy stress tensor σ is a $n \times n$ matrix describing the internal forces.

The linear elasticity equations, assuming unit mass density, are

$$\epsilon = \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) \tag{2.3.1}$$

$$\sigma = C(\epsilon) \tag{2.3.2}$$

$$\nabla \cdot \sigma - \ddot{\boldsymbol{u}} = -\boldsymbol{f}, \tag{2.3.3}$$

where $\boldsymbol{f}$ is the load vector, and C is a tensor product, which will be specified for the relevant dimensions in the following sections. Equation (2.3.1) is a linearization of the definition of strain, Equation (2.3.2) is a material dependent relation between strain and stress, and Equation (2.3.3) is Newton's second law. In this thesis we will use two versions of the equations:

Two-Dimensional Elasticity

In the following we will use subscripts as indices describing indices related to the direction of components, and not as partial derivatives.

[2]Any other constant values of these could have been removed by scaling time and space dimensions.

Define the linear operator

$$\bar{\boldsymbol{\epsilon}}(\boldsymbol{v}) = \begin{bmatrix} \frac{\partial v_x}{\partial x} \\ \frac{\partial v_y}{\partial y} \\ \frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \end{bmatrix}, \tag{2.3.4}$$

the matrix

$$C = \frac{1}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{2.3.5}$$

and the vector

$$\bar{\boldsymbol{\sigma}} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}, \tag{2.3.6}$$

.

Then linear elasticity equations can be written

$$\bar{\boldsymbol{\sigma}} = C\bar{\boldsymbol{\epsilon}}(\boldsymbol{u}) \tag{2.3.7}$$

$$\nabla \cdot \sigma - \ddot{\boldsymbol{u}} = -\boldsymbol{f}, \tag{2.3.8}$$

Note that $\sigma_{xy} = \sigma_{yx}$, so σ (matrix) and $\bar{\boldsymbol{\sigma}}$ (vector) contain the same variables.

Three-Dimensional Elasticity

For the three-dimensional case, we define the linear operator

$$\bar{\boldsymbol{\epsilon}}(\boldsymbol{v}) = \begin{bmatrix} \frac{\partial v_x}{\partial x} \\ \frac{\partial v_y}{\partial y} \\ \frac{\partial v_z}{\partial z} \\ \frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \\ \frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z} \\ \frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \end{bmatrix}, \tag{2.3.9}$$

the matrix

$$C = \frac{1}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}, \tag{2.3.10}$$

and the vector

$$\bar{\boldsymbol{\sigma}} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{yz} \\ \sigma_{zx} \\ \sigma_{xy} \end{bmatrix}. \tag{2.3.11}$$

Equations (2.3.7) and (2.3.8) hold for this case too.

Wellposedness of Initial Boundary Value Problem

In order to have unique solution, we add initial and boundary conditions to the equations. In this thesis we use Dirichlet boundary conditions on all of the boundary. The initial boundary value problem reads

$$\bar{\boldsymbol{\sigma}} = C\bar{\boldsymbol{\epsilon}}(\boldsymbol{u}) \qquad \forall x \in \Omega \tag{2.3.12}$$
$$\nabla \cdot \sigma - \ddot{\boldsymbol{u}} = -\boldsymbol{f} \qquad \forall x \in \Omega \tag{2.3.13}$$
$$u(t,x) = g(t,x) \qquad \forall x \in \partial\Omega \tag{2.3.14}$$
$$u(0,x) = u_0(x) \qquad \forall x \in \Omega \tag{2.3.15}$$
$$\dot{u}(0,x) = w_0(x) \qquad \forall x \in \Omega, \tag{2.3.16}$$

where g is the boundary conditions and u_0 and w_0 the initial conditions.

Proof of uniqueness of the solution of this can be found in [54]. [55] proves existence of solution, and continuous dependence on the data, in the slightly less general situation with $\mathcal{C}^\infty$-smooth functions. This generalizes to our case using density of $\mathcal{C}^\infty$ in Sobolev spaces[56].

2.3.1 Discretization with Finite Elements

We use implicit euler for the time derivatives of $\boldsymbol{u}$. Denote by $\widehat{\boldsymbol{u}}^{(i)}(\cdot)$ the approximation of $\boldsymbol{u}(ik,\cdot)$. The time derivatives are approximated by

$$\ddot{\boldsymbol{u}}(ik) \approx \frac{\dot{\boldsymbol{u}}(ik) - \dot{\boldsymbol{u}}((i-1)k)}{k} \approx \frac{\boldsymbol{u}^{(i)} - 2\boldsymbol{u}^{(i-1)} + \boldsymbol{u}^{(i-2)}}{k^2} \tag{2.3.17}$$

Inserted in Equation (2.3.7), we get

$$\nabla \cdot \sigma^{(i)} - \frac{\boldsymbol{u}^{(i)} - 2\boldsymbol{u}^{(i-1)} + \boldsymbol{u}^{(i-2)}}{k^2} = -\boldsymbol{f}^{(i)} \tag{2.3.18}$$

We denote by Ω the spacial domain, $\partial\Omega$ the boundary, and λ and $\lambda_{\partial\Omega}$ their respective Borel measures. As with the heat equation, we require integrable first derivatives, and use the solution space $X = \times_{i=1}^{d} H^1(\Omega)$, where d is the number of dimensions. We multiply with a test function $\boldsymbol{v} \in X_0$ vanishing on the boundary and integrate over the spacial domain.

$$\int \left(\boldsymbol{v}^T \nabla \cdot \sigma^{(i)} - \boldsymbol{v}^T \frac{\boldsymbol{u}^{(i)} - 2\boldsymbol{u}^{(i-1)} + \boldsymbol{u}^{(i-2)}}{k^2} \right) d\lambda = -\int \boldsymbol{v}^T \boldsymbol{f}^{(i)} d\lambda \tag{2.3.19}$$

We may rewrite the first term (using that $v = 0$ on $\partial\Omega$)

$$\int_\Omega \boldsymbol{v}^T \nabla \cdot \sigma^{(i)} d\lambda = \int_\Omega \sum_{k=1}^{n}\sum_{l=1}^{n} v_k \frac{\partial \sigma_{kl}^{(i)}}{\partial l} d\lambda \tag{2.3.20}$$
$$= \sum_{k=1}^{n}\sum_{l=1}^{n} \left(\int_\Omega -\frac{\partial v_k}{\partial l}\sigma_{kl} d\lambda + \int_{\partial\Omega} v_k \sigma_{kl}^{(i)} d\lambda_{\partial\Omega} \right) \tag{2.3.21}$$
$$= \sum_{k=1}^{n}\sum_{l=1}^{n} \int_\Omega -\frac{\partial v_k}{\partial l}\sigma_{kl}^{(i)} d\lambda \tag{2.3.22}$$
$$= -\int_\Omega \bar{\boldsymbol{\epsilon}}(\boldsymbol{v})^T \bar{\boldsymbol{\sigma}}^{(i)} d\lambda \tag{2.3.23}$$
$$= -\int_\Omega \bar{\boldsymbol{\epsilon}}(\boldsymbol{v})^T C \bar{\boldsymbol{\epsilon}}(\boldsymbol{u}^{(i)}) d\lambda \tag{2.3.24}$$

Inserting into Equation (2.3.19) above we get, after changing sign on both sides,

$$\int \left(\bar{\boldsymbol{\epsilon}}(\boldsymbol{v})^T C \bar{\boldsymbol{\epsilon}}(\boldsymbol{u}^{(i)}) d\lambda + \boldsymbol{v}^T \frac{\boldsymbol{u}^{(i)} - 2\boldsymbol{u}^{(i-1)} + \boldsymbol{u}^{(i-2)}}{k^2} \right) d\lambda = \int \boldsymbol{v}^T \boldsymbol{f}^{(i)} d\lambda. \tag{2.3.25}$$

We are now ready to make the jump to finite dimensional space. For $\boldsymbol{u}^{(i)} \in X^h = \text{span}(\{\boldsymbol{\phi}_m\}_{m=1}^N$ for some finite N, and $\boldsymbol{v} \in X_0^h = X_0 \cup X^h$, let V and $U^{(i)}$ be the vectors such that $\boldsymbol{v} = \sum_{m=1}^N \boldsymbol{\phi}_m V_m$ and $\boldsymbol{u}^{(i)} = \sum_{m=1}^N \boldsymbol{\phi}_m U_m^{(i)}$. Lets define the $N \times N$ matrices $A = (a_{kl})$ and $M = (m_{kl})$ by $a_{kl} = \int_\Omega \bar{\boldsymbol{\epsilon}}(\boldsymbol{\phi_k})^T C \bar{\boldsymbol{\epsilon}}(\boldsymbol{\phi_l}) d\lambda$ and $m_{kl} = \frac{1}{k^2} \int_\Omega \boldsymbol{\phi_k}^T \boldsymbol{\phi_l} d\lambda$, and the vectors $F^{(i)} = (f_k^{(i)})$ by $f_k^{(i)} = \int \boldsymbol{\phi}_k^T \boldsymbol{f}^{(i)} d\lambda$. We can then write the above equation as

$$V^T A U^{(i)} + V^T M U^{(i)} - 2V^T M U^{(i-1)} + V^T M U^{(i-2)} = V^T F^{(i)}. \tag{2.3.26}$$

We use the lifting function approach to handle the boundary conditions. Rearranging the terms, we end up with the equations

$$(A + M)U^{(i)} = F^{(i)} + 2MU^{(i-1)} - MU^{(i-2)}, \tag{2.3.27}$$

which is solved iteratively by some appropriate numerical scheme.

2.4 DDM Using Neural Networks

DDM has become very popular in recent years, following the increased availability of large amounts of data and computational resources as well as free, state of the art machine learning libraries. Contrary to PBM, DDM learns from data, and assuming the data provided is of sufficient quantity as well as quality, it will learn the full physics of the system it describes, independent of our knowledge and understanding of it. Some of the advantages of this approach is highly computationally efficient predictions, ability to adapt to changes on the fly, as well as high accuracy even for difficult problems, when sufficient data is available. Nevertheless, they are not widely used in high stake situations due to weaknesses like poor generalizability, lack of explainability and theory of error analysis and stability, and the need for large amounts of data. Although there is a large variety of DDM available we use deep neural networks (DNN) in this thesis, as they are universal approximators [57, 58, 59] and hence have the ability to model highly complex nonlinear phenomena.
(The paragraph above was adapted from [45])

A DNN with n layers is a function $D : \mathbb{R}^{d_0} \to \mathbb{R}^{d_n}$ defined by

$$D(x) = T_n \circ s_{n-1} \circ T_{n-1} \circ s_{n-2} \cdots \circ s_1 \circ T_1(x) \tag{2.4.1}$$

where $T_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ are affine transformations and $s_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$ are some preferably nonlinear activation functions.[3] Each transformation T_i is determined by a weight matrix and a bias vector, a total of $d_{i-1} \cdot d_i + d_i$ values. These values are tuned to minimize the error on the training data.

DNNs are widely used, due to their simplicity and yet astonishing performance in many situations. They can be applied to a large variety of problems, and have delivered impressive achievements across numerous fields of research and applications. Meanwhile, they also have some inherent weaknesses. They are tuned for performing on the training data. While there are many ways of preventing the network from being too specialized [60], they will not be good at extrapolation cases where the prediction task is somehow qualitatively different

[3] With linear activation functions the method reduces to multivariate linear regression. For the universal approximation property it should also be nonpolynomial.

from the training tasks. In addition, they are not easily explainable or predictable, meaning its hard to explain what kind of patters the network will look for, and unexpected predictions can occur. Compared to PBMs, DDMs usually make predictions much faster, but may need long training times for optimal performance.

2.5 Corrective Source Term Approach

In this section we present a brief justification of the use of CoSTA. It is based on the more thorough argument that can be found in [45].

Consider the differential equation

$$\begin{aligned} L_\Omega u &= f, \quad \forall\, x \in \Omega \\ L_{\partial\Omega} u &= g, \quad \forall\, x \in \partial\Omega, \end{aligned} \tag{2.5.1}$$

where $L_\Omega, L_{\partial\Omega}$ are differential operators, f is a source term and g a function specifying the boundary condition. Now let $\widetilde{u}$ be the solution to the perturbed problem

$$\begin{aligned} \widetilde{L}_\Omega \widetilde{u} &= \widetilde{f}, \quad \forall\, x \in \Omega \\ \widetilde{L}_{\partial\Omega} \widetilde{u} &= \widetilde{g} \quad \forall\, x \in \partial\Omega, \end{aligned} \tag{2.5.2}$$

where the perturbations $\widetilde{\cdot}$ are due to imperfections such as unknown physics, modelling errors, discretization error, or inaccurate data. For example, we will later in this thesis simplify a nonlinear operator L_Ω with a linear (and discretized) $\widetilde{L}_\Omega$. Assume we can calculate the residuals defined as

$$\begin{aligned} r_\Omega &= \widetilde{L}_\Omega(u - \widetilde{u}) \\ r_{\partial\Omega} &= \widetilde{L}_{\partial\Omega}(u - \widetilde{u}), \end{aligned} \tag{2.5.3}$$

and let $\widehat{\widetilde{u}}$ be the solution of the corrected, perturbed problem

$$\begin{aligned} \widetilde{L}_\Omega \widehat{\widetilde{u}} &= \widetilde{f} + r_\Omega, \quad \forall\, x \in \Omega \\ \widetilde{L}_{\partial\Omega} \widehat{\widetilde{u}} &= \widetilde{g} + r_{\partial\Omega} \quad \forall\, x \in \partial\Omega. \end{aligned} \tag{2.5.4}$$

Using the definition of the residuals, as well as the perturbed Equation (2.5.2), we see that

$$\begin{aligned} \widetilde{L}_\Omega \widehat{\widetilde{u}} &= \widetilde{f} + \widetilde{L}_\Omega(u - \widetilde{u}) = \widetilde{L}_\Omega u, \quad \forall\, x \in \Omega \\ \widetilde{L}_{\partial\Omega} \widehat{\widetilde{u}} &= \widetilde{g} + \widetilde{L}_{\partial\Omega}(u - \widetilde{u}) = \widetilde{L}_{\partial\Omega} u \quad \forall\, x \in \partial\Omega, \end{aligned}$$

which reduces to $\widehat{\widetilde{u}} = u$ if the corrected, perturbed problem (2.5.4) yields a unique solution. From this argument, we see that the source term corrections are able to compensate for perturbations in the differential operators as well as the source terms.

In real scenarios, we obviously cannot calculate the residual exactly, as that would require the solution we are trying to estimate. The idea of the CoSTA method is to use a DDM to estimate the residual. For the input of the DDM we use the uncorrected solution $\widetilde{u}$. This means the PBM is used twice, first to solve Equation (2.5.2) for $\widetilde{u}$, then Equation (2.5.4) for $\widehat{\widetilde{u}}$. Since the DDM output is but a correction, the CoSTA framework exploits the knowledge in the physical model, unlike a pure DDM approach.

For the discretized version of heat equation derived in section 2.2.1, the corrected equation is

$$(M + kA)(\boldsymbol{u_i}) = M\boldsymbol{u_{i-1}} + F_i + r. \tag{2.5.5}$$

Likewise, the corrected equation for elasticity reads

$$(A + M)U^{(i)} = F^{(i)} + 2MU^{(i-1)} - MU^{(i-2)} + r. \tag{2.5.6}$$

As the boundary values are known, we do not need any correction for these elements. Therefore the DDM only need to output values for the inner nodes.

In combining a PBM and a DDM, CoSTA aims to combine the strengths of both of these, but it is clear that it also inherits weaknesses from both. It requires both a physical description of the system it models, as well as empirical data. It needs to be trained before it can be used, but it also uses much computational resources to make a prediction, as two steps must be made with the PBM for every step with the CoSTA, in addition to the DDM prediction. On the other hand, it is affected less by inaccuracy in the physical description than a PBM, and due to the minor role of the DDM part compared to a pure DDM method, it should also be less affected by poor quality or quantity of data. Finally, the extra accuracy it has shown easily justifies the additional computational cost.

2.6 Method of Manufactured Solutions

We wish to evaluate the performance of the models in a variety of different scenarios. To this end we use the method of manufactured solutions [61], which involves choosing a solution u, and calculating the source f from the governing equation, before trying to reproduce the solution using f, $u|_{t=0}$ and $u|_{\Omega}$. The alternative is to choose the conditions f, $u|_{t=0}$ and $u|_{\Omega}$ to use for approximating the solution $\widetilde{u}$. But then the correct solution is unknown, so the error $u - \widetilde{u}$ must be approximated by using a more precise method [4], which is very computationally demanding, and only gives an approximate error. The method of manufactured solutions enables easy comparison with the known exact solution to accurately quantify the error of the model.

[4] usually high fidelity numerical solutions of the equation, with a fine grid

Chapter 3
Methodology

This project includes several experiments, to test the usefulness of CoSTA in several situations. An overview of the experiments is shown in Table 3.0.1. In this chapter, we first present the general setup of the experiments in Section 3.1, then present the details of each experiment in the next sections. Unless otherwise stated, all the experiments that follow will use the method and the parameters presented in Section 3.1. What separates the experiments are the PDEs solved, the existence and nature of any modelling errors, and the choice of manufactured solutions.

The design of the experiments, along with choice of parameters for the models, is based on the paper originally introducing CoSTA [45]. For the exact implementation used for the experiments and figures in this project, see `https://github.com/sondsorb/FEM_CoSTA`.

3.1 General Setup

3.1.1 Equation and Manufactured Solution

The exact solution follows a PDE

$$L(u) = f, \tag{3.1.1}$$

where u is the solution, f is a source term, and L is some possibly nonlinear differential operator. We refer to this equation as the governing equation.

We model this using three approaches, PBM, CoSTA and DDM, and then compare the results. The PBM and CoSTA will use a finite element discretization of the governing equation, possibly with some modelling errors, of the form

$$\widehat{L}\widehat{u} = f, \tag{3.1.2}$$

where $\widehat{L}$ is some linear, finite dimensional differential operator approximating L, and $\widehat{u}$ is the approximation of the solution u.

The solution u is chosen, and the source term f derived by inserting u in the governing equation. The source term, along with appropriate initial and boundary conditions, are used

Table 3.0.1: Description of experiments and modelling errors (in addition to discretization error).

Experiment	Modelling error
Heat with reduced dimensionality, 2D-1D	A dimension is ignored
Heat with reduced dimensionality, 3D-2D	A dimension is ignored
Slightly nonlinear heat	PDE is linearized
More nonlinear heat	PDE is linearized
Linear elasticity with source term	None
Linear elasticity without source term	Source term replaced with zero
Linear elasticity with reduced dimensionality	A dimension is ignored
Nonlinear elasticity	PDE is linearized

Table 3.1.1: Values of the parameter α used for training, validation and testing.

Set usage	Notation	Values
Testing	$\mathcal{A}_{\text{test}}$	$\{-0.5, 0.7, 1.5, 2.5\}$
Validation	$\mathcal{A}_{\text{val}}$	$\{0.8, 1.1\}$
Training	$\mathcal{A}_{\text{train}}$	$\{0.1, 0.2, ...2.0\} \backslash (\mathcal{A}_{\text{test}} \cup \mathcal{A}_{\text{val}})$

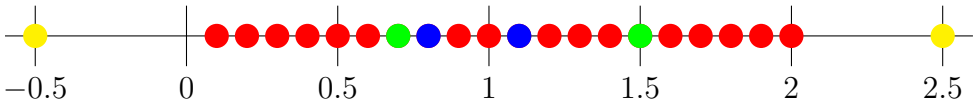


Figure 3.1.1: Values of α used for ● training, ● validation, ● interpolation testing and ● extrapolation testing.

as input for the different methods to reproduce an approximate solution $\widehat{u}$. This process is repeated for several different choices of solution u as an attempt to avoid misrepresenting the usability of the method outside the test scenarios. The manufactured solutions include a variety of polynomial, exponential and harmonic functions as an attempt to test the methods on a broad class of solutions. The set of manufactured solution used in each experiment are presented in their respective sections.

3.1.2 Experiment Procedure

In order to use DDM, either alone or in the CoSTA framework, it is necessary to have some training data that the models can use to learn from. In the real world, this may be measurements from previous situations that are more or less similar to the one being modelled, or from high fidelity simulations of possible outcomes, if there is sufficient computational resources available in the offline phase of the model. To imitate this, the solutions u, and in some experiments also the operator L, are designed to depend on the parameter α. We use one set of values of this parameter for training, one for validation and one for testing, as is presented in Table 3.1.1 and visualized in Figure 3.1.1. The test set $\mathcal{A}_{\text{test}}$ includes the two values 0.7 and 1.5 which are close on both sides to values in the training set. These situations should be very similar to the training data, and will therefore be referred to as interpolation scenarios. The test set also contain -0.5 and 2.5, which should produce solutions less similar to the training ones. Machine learning models are known to perform good for tasks similar to the training data, but much worse on less similar ones, and CoSTA and DDM are expected to perform worse on these values, which will be referred to as extrapolation scenarios. This choice of values makes it possible to observe both the accuracy and generalizability of the methods. The negativity of the value -0.5 might produce solutions qualitatively different from the ones with positive α, although this depends on the manufactured solutions.

The process of the experiments is presented in Algorithm 1. K denotes the total number of time steps. Note the difference between $\breve{U}$, $\widetilde{U}$ and $\widehat{U}$: we denote by $\breve{U}$ the vector form of the exact solution[1]. The predictions $\widetilde{U}$, used for training, are based on the previous exact step, while $\widehat{U}$, used for testing, are based on the previous predicted step (the first predicted step is based on the exact initial values). As in Chapter 2, we use the superscript to denote the time step, e.g. $\widehat{U}^{(i)}$ is at time $t_i = ik$.

The methods map the prediction $\widehat{U}^{(i-1)}$to $\widehat{U}^{(i)}$. This map is used iteratively to calculate $\widehat{U}^{(K)}$ at the final time step, from the initial (exact) input $\breve{U}^{(0)}$ [2]. We should therefore expect

[1]That is, the piecewise linear function characterised by $\breve{U}$ equals the exact solution on the grid points

[2]Note that there are several other alternative ways to solve such a multistep time series forecasting problem,

Algorithm 1: Pseudocode showing how the experiments were performed. Details on how training and evaluations were done are omitted, see Sections 3.1.3 and 3.1.4 for this.

Pick a solution $u_{exact}(t, x, y, z, \alpha)$;
Use governing equation(s) (the PDE before simplification) to calculate f from u_{exact};
for $\alpha \in \mathcal{A}_{train}$ *and* $\alpha \in \mathcal{A}_{val}$ **do**
  for $i = 0, 1, 2...K-1$ **do**
    Use (simplified) PBM to calculate $\widetilde{U}^{(i+1)}_{\text{PBM}} = \text{PBM}(\breve{U}^{(i)})$;
    Calculate the residual $\widetilde{r}^{(i+1)} = \widehat{L}(\breve{U}^{(i+1)} - \widetilde{U}^{(i+1)}_{\text{PBM}})$;
Train DDM to map $\breve{U}^{(i)}$ to $\breve{U}^{(i+1)}$, i.e. minimize
  $\sum_{\alpha \in \mathcal{A}_{\text{train}}} \sum_{i=0}^{K-1} \left| \breve{U}^{(i+1)} - \text{DDM}(\breve{U}^{(i)}) \right|^2$;
Train CoSTA network to map $\widetilde{U}^{(i+1)}_{\text{PBM}}$ to $\widetilde{r}^{(i+1)}$, i.e. minimize
  $\sum_{\alpha \in \mathcal{A}_{\text{train}}} \sum_{i=0}^{K-1} \left| \widetilde{r}^{(i+1)} - \text{DDM}_{\text{CoSTA}}(\text{PBM}(\breve{U}^{(i)})) \right|^2$;
$\widehat{U}^{(0)}_{\text{DDM}} = \breve{U}^{(0)}$;
$\widehat{U}^{(0)}_{\text{PBM}} = \breve{U}^{(0)}$;
$\widehat{U}^{(0)}_{\text{CoSTA}} = \breve{U}^{(0)}$;
for $\alpha \in \mathcal{A}_{test}$ **do**
  for $i = 0, 1, 2...K-1$ **do**
    calculate $\widehat{U}^{(i+1)}_{\text{DDM}} = \text{DDM}(\widehat{U}^{(i)}_{\text{DDM}})$;
    calculate $\widehat{U}^{(i+1)}_{\text{PBM}} = \text{PBM}(\widehat{U}^{(i)}_{\text{PBM}})$;
    calculate $\widehat{U}^{(i+1)}_{\text{CoSTA}} = \text{PBM}(\widehat{U}^{(i)}_{\text{CoSTA}}, \text{DDM}_{\text{CoSTA}}(\text{PBM}(\widehat{U}^{(i)}_{\text{CoSTA}})))$;
Evaluate by comparing to exact solution u_{exact}

that any error in one step propagates into the next. It is not feasible to train DDM or CoSTA to counteract this by fixing the global error, the most obvious reason for this being that the global error is not available until the training is finished. Therefore, the exact solution is used as both the input and output training data in every step.

For testing scenarios, the exact solution at time $t = 0$ is used as the input for the first step, and then each model predict each step based on the previous. Boundary conditions are also used as input for each step, along with the source function values. The error is measured at each step, and presented in the result chapters.

For actual usage of this method for prediction, the first two lines and the last line of Algorithm 1 is skipped. Instead, f must be known (or approximated), along with u_{exact} in training scenarios, and the initial conditions and boundary conditions in the testing scenarios.

3.1.3 Methods

PBMs

For the PBMs, both standalone PBMs and in CoSTA, a linear FEM model is used, with first order (piecewise linear) Lagrange elements [52] on an equidistant grid. The time interval is divided into time steps of constant length $k = (\text{Number of time steps})^{-1}$. Unless specified otherwise, the spatial domain used is the unit interval $[0, 1]$ or unit square $[0, 1] \times [0, 1]$, and

that does not necessarily suffer from the disadvantages discussed here, although they most definitely have their own ones.

Table 3.1.2: Overview of spacial and temporal resolution.

Experiment	Elements	Time Steps
Heat with reduced dimensionality, 2D-1D	20	5000
Heat with reduced dimensionality, 3D-2D	20×20	2500
Slightly nonlinear heat	20	5000
More nonlinear heat	20	5000
Linear elasticity with source term	15×15	1000
Linear elasticity without source term	15×15	1000
Linear elasticity with reduced dimensionality	15×15	1000
Nonlinear elasticity	10×10	500

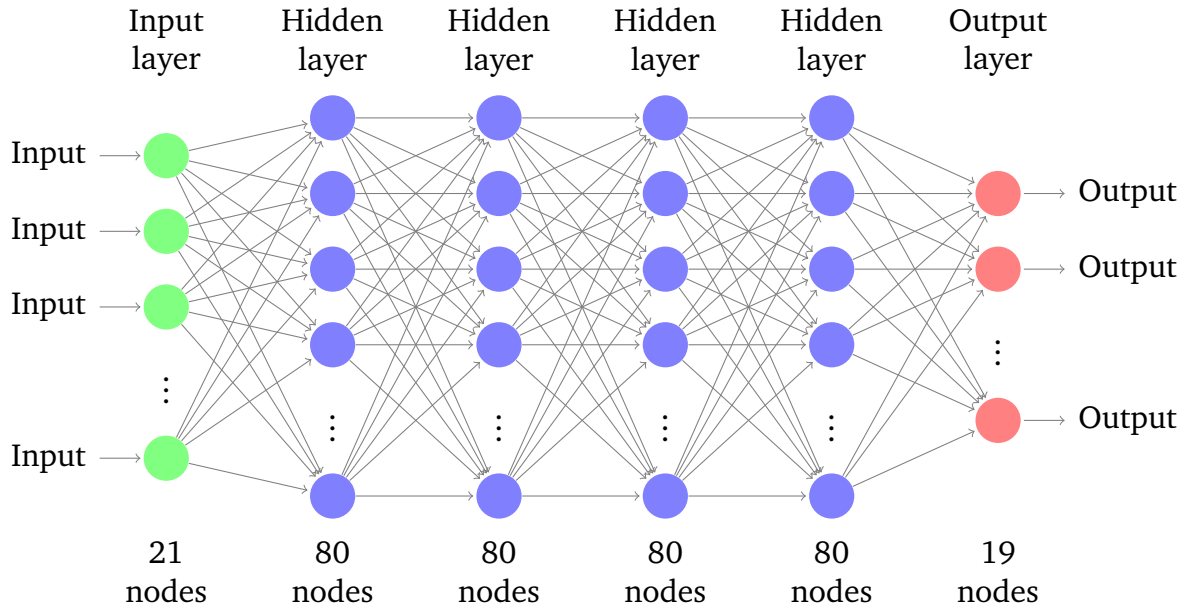


Figure 3.1.2: Visualization of the DNN architecture for the experiments with 20 elements. The nodes represent input, output and intermediate values, while the arrows going between them represent dependencies. Generally there is one input node for each basis function, and one output node for each basis function not on the edge.

time domain is the unit interval $[0, 1]$. The amount of time steps vary between 500 and 5000, and the amount of elements in each spacial dimension vary from 10 to 20. The exact numbers are shown in Table 3.1.2.

Neural Network Architecture and Parameters

The DDMs, both as a model on its own and as part of CoSTA, are deep neural networks, with four dense hidden layers of 80 nodes each. The length of the input and output layers depend on the PDE and discretization. The input vector contain every basis function, while the output does not contain the functions on the boundary. For the one-dimensional heat equation with 20 elements, this gives 21 input nodes and 19 output nodes, as shown in Figure 3.1.2. Since the elasticity equation has a vector field solution with twice as many basis functions for two-dimensional solutions, the length of the input and output layers are multiplied accordingly. This architecture is used for every neural network - both in DDM and CoSTA.

The neural networks are implemented using TensorFlow [62]. As activation function we use leaky ReLU [63], with coefficient 0.01 for negative inputs. Training parameters are pre-

Table 3.1.3: Parameters used for the training procedures for all of the neural networks.

Loss function	MSE
Optimizer	Adam [64]
Learning rate	$1e-5$
Patience	20

sented in Table 3.1.3. A patience of 20 means the training stops when the score on the *validation* set has not improved for the last 20 optimizer steps. All the data is normalized before it is inputted in the DNN, and the output is unnormalized, based on the training data.

The architecture and parameters used for the DNNs are not tuned for best performance on the problems used in this paper, but rather taken from [45], who tuned them briefly for some different heat modelling problems.

3.1.4 Evaluation and Visualization

To evaluate the performance of the different methods, relative root mean square error (RRMSE) is measured at each time step. This value is defined

$$\text{RRMSE}(U^{(i)}, \breve{U}^{(i)}) = \frac{||U^{(i)} - \breve{U}^{(i)}||_2}{||\breve{U}^{(i)}||_2} \tag{3.1.3}$$

for a prediction U and correct solution $\breve{U}$. For models with stochastic results (i.e. DDM and CoSTA, that includes a DNN with random initialisation), 10 models are trained and used. In a real world application, the objective would determine if we are interested in the least amount of error at all time steps, or only at the last one. In this thesis we are interested in both. In the result plots presented in Chapter 4, the mean of the RRMSE of the 10 models is plotted as a line on a logarithmic scale, as a function of time steps. The uncertainty of this mean is also calculated[3], and visualized in the plots by shading the area between the mean RRMSE, and the mean RRMSE plus one standard deviation. Mean RRMSE *minus* one standard deviation is not plotted since it might be negative, which does not work well on logarithmic scales. Plots of the mean of the final solution is also plotted, and plus/minus the uncertainty around this mean is shaded, for one-dimensional solutions. For experiments with more than one dimension, plots of the final solutions take up a lot of space, without being very interesting, and is therefore put in the appendix.

3.2 Modelling Heat with Reduced Dimensionality

With each added physical dimension in a model, the computational complexity increases greatly. For this reason, if the situations allows it, reducing the dimensionality of a problem can greatly reduce the cost of solving it. In this section we model a one-dimensional line in a two-dimensional object by ignoring the effects of the second dimension, and compare to using a two-dimensional PBM of the same object. Likewise, we also model a two-dimensional plane of a three-dimensional object by ignoring the effects of the third dimension. Of course, neglecting dimensions, at least using PBMs, will likely cause large errors, depending on the scenario. We will test if CoSTA can correct for such errors.

[3]The standard deviation is calculated using one reduced degree of freedom due to the estimation of the mean.

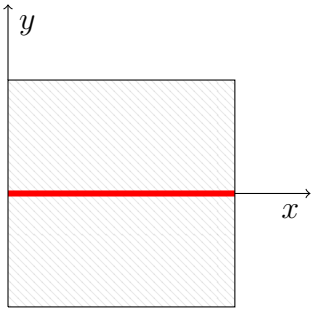


Figure 3.2.1: The object modelled in the first dimensional reduction setup. We predict the temperature only at the thick red line at $y = 0$. Boundary conditions are needed at the end points of this line.

For the first setup, we want to predict the temperature development of the two-dimensional square object at $[0, 1] \times [-0.5, 0.5]$, but we only need the temperature at the line at $y = 0$, as seen in Figure 3.2.1. This can be modelled by a two-dimensional PBM, based on the governing equation, which is the two-dimensional heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f. \tag{3.2.1}$$

The term $\frac{\partial^2 u}{\partial y^2}$ requires the PBM to model the two-dimensional area above and below the line $y = 0$, all the way to the known boundary conditions. In this experiment, we instead use the one-dimensional heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f, \tag{3.2.2}$$

to make the PBM, and train the DMM part of CoSTA to correct for this simplification. This allows us to only include the one-dimensional elements at $y = 0$ in the model, which will result in a much faster prediction. It also requires less boundary conditions (only the end points of the line, instead of the boundary of the square) and initial conditions (the line instead of the plane). Note that while the method only uses temperatures at the one-dimensional domain, the actual solution will depend heavily on what happens in the rest of the two-dimensional object. Any extra knowledge of the system, like boundary conditions on the edges, or two-dimensional initial conditions, can thus be used as an indication of similarity between testing and training data.

For the second setup, we want to predict the temperature development of a 3-dimensional object, the cube $[0, 1] \times [0, 1] \times [0, 1]$, and we are only interested in the temperature at one of the sides, $[0, 1] \times [0, 1] \times \{0\}$. This setup is illustrated in Figure 3.2.2. We use the two-dimensional heat equation in a PBM to predict the temperature distribution of this side of the cube. This setup differs from the first one not only in number of dimensions, but also in the nature of the problem: Unlike in the first setup, the lower-dimensional domain of interest is at the boundary of the full object. As we ignore this dimension anyways, this difference has no effect on our method, but it illustrates a different usage of this method. Here we are not able to use a three-dimensional PBM to solve the problem, as the boundary conditions at the lower-dimensional domain are unknown by assumption. The two-dimensional PBM, on the other hand, only require initial conditions at the plane, and boundary conditions at the edges of this plane. This kind of initial conditions are much more available than the full three-dimensional ones, as they can easily be measured by a thermal camera. In situations where the temperature of the whole object is of interest, this method might be useful for predicting the boundary conditions, to then use in a full-dimensional model.

The manufactured solutions used in this experiment are presented in Table 3.2.1. The same solutions are used for both setups, with $z = 0$ in the first one. Figure 3.2.3 shows the

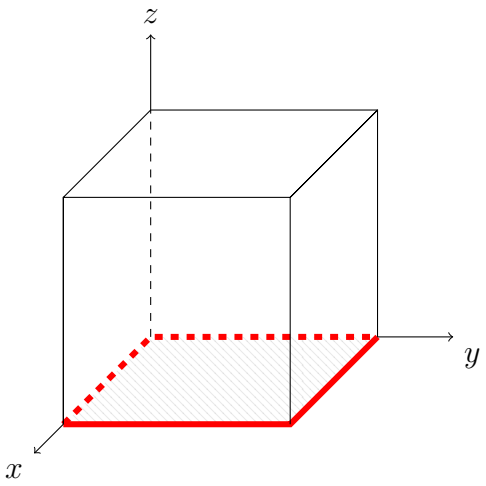


Figure 3.2.2: Example of an object to model using the dimensional reduction method. While the full model is a three-dimensional cube, we only predict the shaded side. Boundary conditions needed are those at the thick red edges at $z = 0$.

Table 3.2.1: Manufactured solutions the for reduced dimensionality heat experiments.

Label	$u(t, x, y, z, \alpha)$
d1	$\sin(4\alpha t + x + \alpha y + 2z)$
d2	$(\cos(\alpha x) + \sin(\alpha + z) + \sin(t + y))(e^{-t} + 2e^{t-1})$
d3	$5t\alpha(x + y - z)^2 + 5\sqrt{t}2^{x-y+z-\alpha}$
d4	$\frac{1}{1+x+y+z} + \frac{\ln(2+\alpha+x+y+z)}{\sqrt{5t+1}}$

manufactured solutions at $y = z = 0$, for three values of α. We see that solution *d1* is more greatly effected by the value of α than the other solutions, and for high values of α it also has a stronger dependency on t. For these reasons we expect DDM and CoSTA to struggle more with this solution.

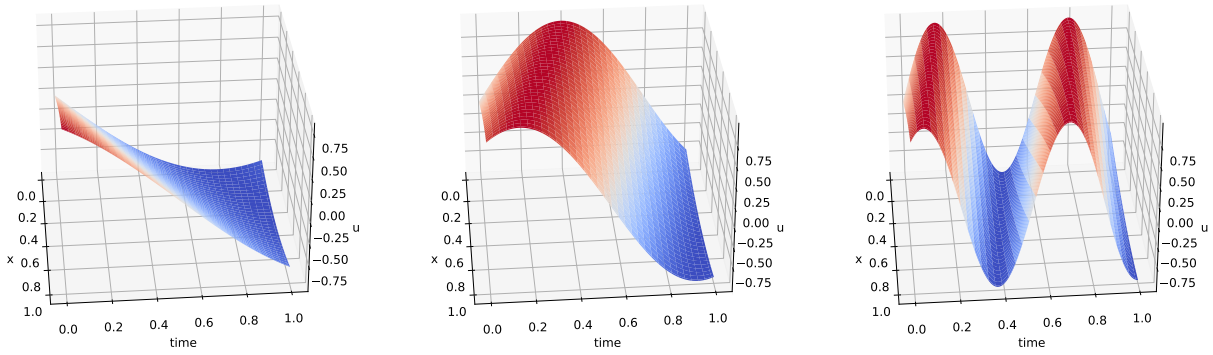
For the first setup, a two-dimensional PBM was used in addition to the one-dimensional models, in order to see how well the one-dimensional CoSTA corrects for the ignored dimension. It uses 20×20 elements, so the solution space at $y = 0$ matches the one-dimensional methods. It is evaluated only at the this line, like the other methods.

3.3 Modelling Nonlinear Heat

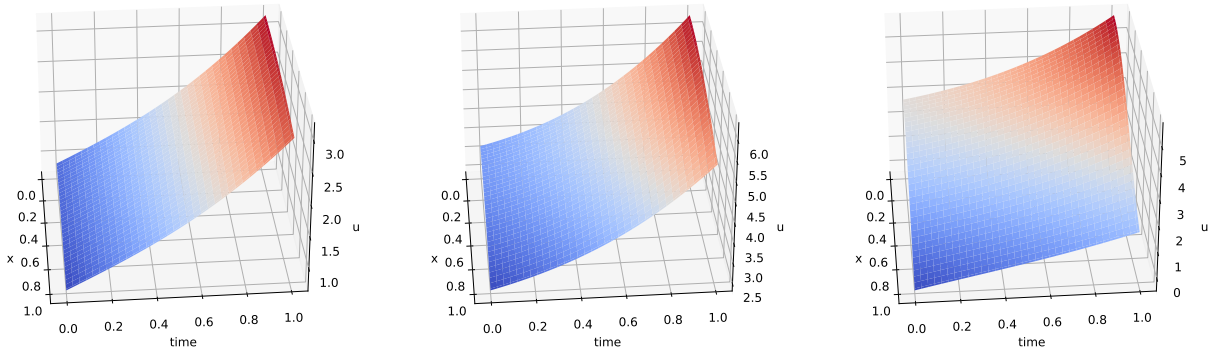
In this section we use CoSTA on a nonlinear heat equation, with a linear PBM, to test how well the CoSTA term corrects for this modelling error.

The conductivity of a material generally depends on the temperature. The linear PDE $u_t = \nabla u + f$ is therefore only a good approximation when the temperature is somewhat constant, so that the conductivity may be described by a constant value. For a more accurate and general equation, we must use $k = k(u)$ (where the expression of $k(u)$ is material dependent), and the PDE

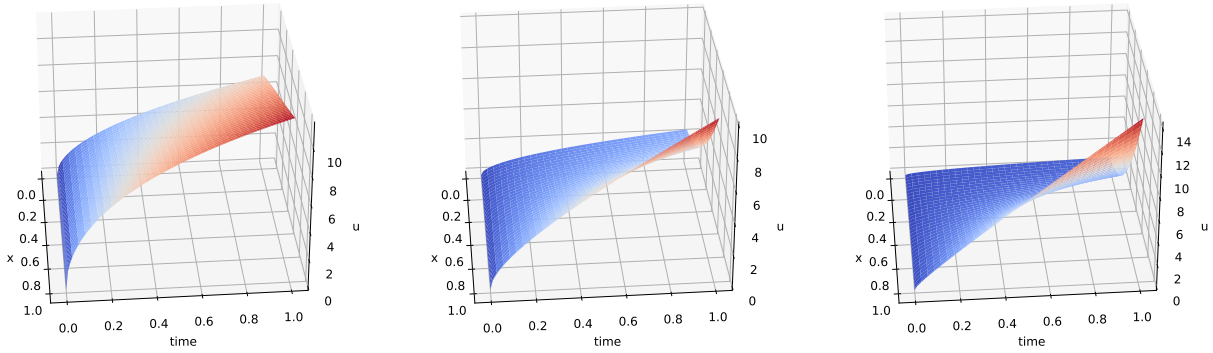
$$u_t = \nabla(k\nabla u) + f \tag{3.3.1}$$
$$= \nabla k \cdot \nabla u + k\Delta u + f, \tag{3.3.2}$$



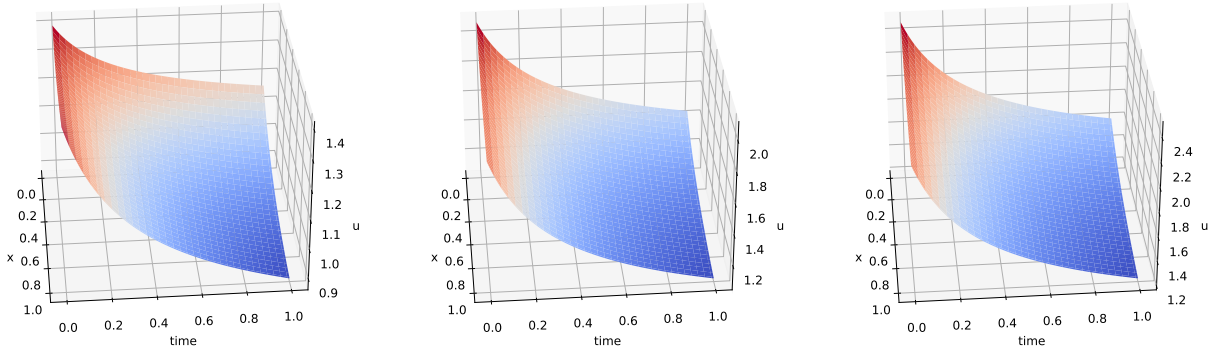
(a) *d1*$(\alpha = -0.5)$ **(b)** *d1*$(\alpha = 1)$ **(c)** *d1*$(\alpha = 2.5)$



(d) *d2*$(\alpha = -0.5)$ **(e)** *d2*$(\alpha = 1)$ **(f)** *d2*$(\alpha = 2.5)$



(g) *d3*$(\alpha = -0.5)$ **(h)** *d3*$(\alpha = 1)$ **(i)** *d3*$(\alpha = 2.5)$



(j) *d4*$(\alpha = -0.5)$ **(k)** *d4*$(\alpha = 1)$ **(l)** *d4*$(\alpha = 2.5)$

Figure 3.2.3: Manufactured solutions used for modelling heat with reduced dimensionality, at $y = z = 0$, for $\alpha \in [-0.5, 1, 2.5]$.

Table 3.3.1: Manufactured solutions the for the experiments with temperature dependent conductivity. The first four are for the setup with low nonlinearity, the next four for the setup with high nonlinearity.

Label	$u(t, x, \alpha)$	$k = k(u, \alpha)$
c1	$\sin(5\alpha t + x) + \sin(\alpha x)/2$	$1 + \sin(2\alpha u)/2$
c2	$\cos(\alpha x)(e^{-t} + 2e^{t-1})$	$1 + \frac{\alpha u}{10}$
c3	$\alpha x^2 + 2tx - 2t - 2x + 1$	$e^{u/10}$
c4	$\frac{1}{1+x} + \frac{\alpha+x}{\sqrt{5t+1}}$	$e^{-u/10}$
xc1	$\sin(5\alpha t + x) + \sin(\alpha x)/2$	$1 + \sin(2\alpha u)$
xc2	$\cos(\alpha x)(e^{-t} + 2e^{t-1})$	$1 + \frac{\alpha u}{2}$
xc3	$\alpha x^2 + 2tx - 2t - 2x + 1$	$e^{u/4}$
xc4	$\frac{1}{1+x} + \frac{\alpha+x}{\sqrt{5t+1}}$	$e^{-u/4}$

or for one-dimensional situations, which is what we will be considering in this project,

$$u_t = k_x u_x + k u_{xx} + f. \tag{3.3.3}$$

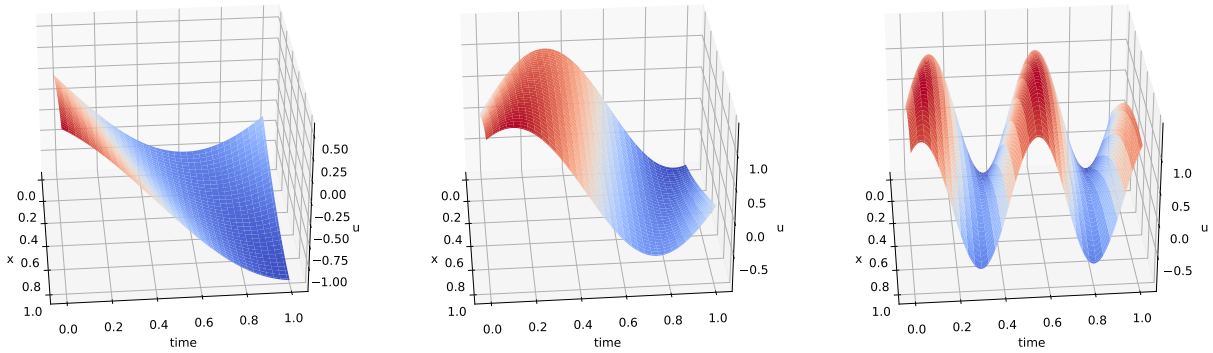
This is a nonlinear PDE. This can not be solved by the linear FEM method used in this project, but need other methods like the Newton-Raphson method [65], which iteratively solves linearized versions of the PDE. This naturally involves a leap in computational complexity from the linear PDE. In this project we attempt to solve this nonlinear PDE using CoSTA, in which the PBM is linear FEM. By this, we aim to have the DDM in the CoSTA learn to correct for the nonlinearity of the PDE. The solution u is chosen, and used in the nonlinear heat equation to derive the source term f, which is used as input for the linear PBM model.

The (constant) value approximating the conductivity should obviously be chosen close to the correct conductivity. In real world applications, the exact conductivity as a function of temperature may not be known, but even if it is, the temperature is by assumption unknown a priori, so the choice of k has to be a more or less educated guess. In this experiment, as we choose the correct conductivity anyways, these are chosen so that the conductivity $k = 1$ is a somewhat good approximation, and we use this in the PBM.

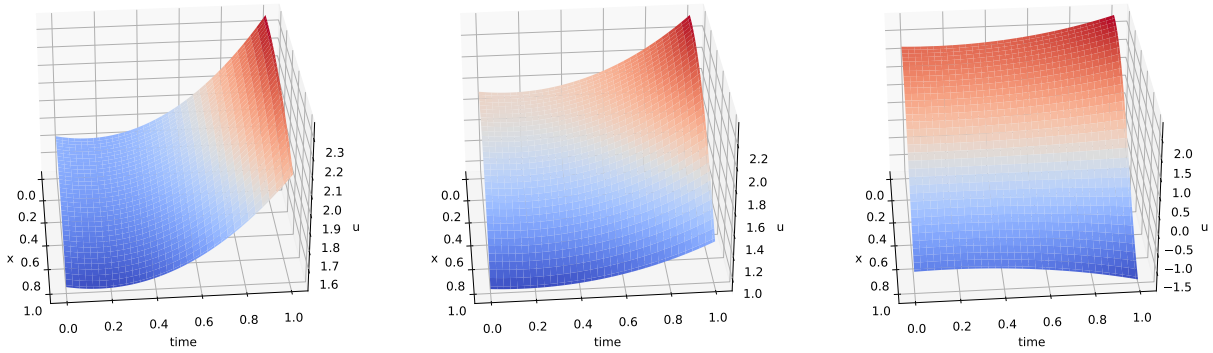
Two experiments are conducted with nonlinear heat, with the same set of manufactured solutions, but with different conductivities. In the first one, the conductivities are relatively close to 1, while in the second the conductivities are more varying. This way we can compare and see how the governing PDEs "closeness" to a linear PDE affects the results. The manufactured solutions and temperature dependent conductivities are found in Table 3.3.1. The solutions are visualized in Figure 3.3.1. They are inspired by the solutions in the previous section, and we again see the first solution *c1* vary the most along t and α, so we expect this to be harder than the other solutions for DDM and CoSTA. 5000 time steps and 20 elements are used on both setups in this chapter.

3.4 Modelling Linear Elasticity With and Without Known Source Term

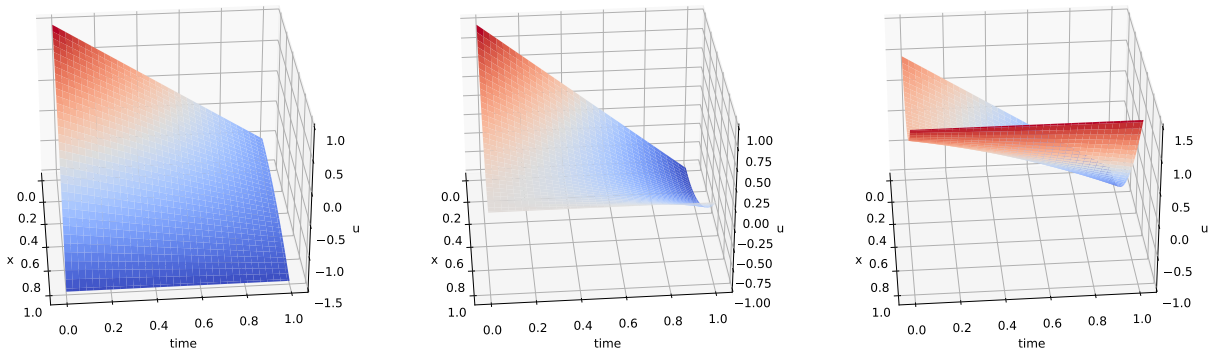
In this section we use the models on a system governed by the two-dimensional transient linear elasticity equation. Manufactured solutions are presented in Table 3.4.1. First, we solve



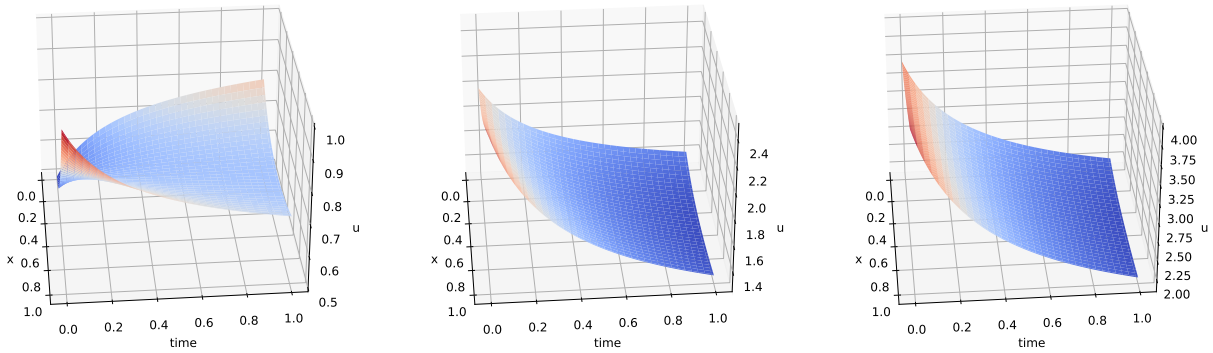
(a) *c1*$(\alpha = -0.5)$ **(b)** *c1*$(\alpha = 1)$ **(c)** *c1*$(\alpha = 2.5)$



(d) *c2*$(\alpha = -0.5)$ **(e)** *c2*$(\alpha = 1)$ **(f)** *c2*$(\alpha = 2.5)$



(g) *c3*$(\alpha = -0.5)$ **(h)** *c3*$(\alpha = 1)$ **(i)** *c3*$(\alpha = 2.5)$



(j) *c4*$(\alpha = -0.5)$ **(k)** *c4*$(\alpha = 1)$ **(l)** *c4*$(\alpha = 2.5)$

Figure 3.3.1: Manufactured solutions of used for nonlinear heat modelling, with $\alpha \in [-0.5, 1, 2.5]$.

Table 3.4.1: Manufactured solutions for elastic problems.

Label	$u(t, x, y, \alpha)$
e1	$\begin{bmatrix} \sin(\pi(x+\alpha y))\cos(\alpha t) \\ \cos(\pi(x+\alpha y))\sin(\alpha t) \end{bmatrix}$
e2	$\begin{bmatrix} \exp(\frac{(-tx^2+y^2)}{(1+\alpha+t^2)}) \\ \exp(\frac{(+tx^2-y^2)}{(1+\alpha+t^2)}) \end{bmatrix}$
e3	$\begin{bmatrix} x^3 + y^2(t+0.5)^{1.5} + xy\alpha \\ x^2 + y^3(t+0.5)^{1.1} + xy\alpha \end{bmatrix}$

the problems without adding any modelling error. For the second test, instead of using the correct source term found from inserting the manufactured solution u in the governing equation and solving for f, we assume the source is unknown and set it to $f = 0$[4]. This will affect the PBM, and thus CoSTA, although not the DDM since this does not use the source term as an input. This will show if CoSTA works as intended and learns to correct the source term. The DDMs do not utilize the source term, so the same results could have been used in both tests. Despite this, new DDMs were trained in both tests, resulting in slightly varying DDM results between the tests, although we will see that the difference turned out to be minimal.

3.5 Modelling Linear Elasticity with Reduced Dimensionality

In this section we try using CoSTA for dimensional reduction on the linear elasticity equation. This is done in the same way as with the second setup with the heat equation, i.e. by making a PBM based on the two-dimensional linear elasticity equation, even though the system is governed by the three-dimensional one. However, since the displacement u is a vector field with as many components as there are dimensions, this method can only predict two of the components of u, being the ones present in the reduced system.[5] Hence the approach is mostly relevant when displacement in the reduced direction is either small and/or of little interest compared to the others. Only the two components predicted are used when calculating the error in this section. Manufactured solutions are presented in Table 3.5.1.

3.6 Modelling Nonlinear Elasticity

In this section we use CoSTA to correct for using a linear FEM solver on solutions of the two-dimensional elasticity equation with nonlinear dependency between strain and stress.

[4] In real scenarios one should use the best available estimation of the correct source

[5] An alternative approach to dimensional reduction for the linear elasticity equation (and other vector field PDEs), is to base the PBM on the three-dimensional model, but replace the derivatives in the ignored z-direction with zero (or some other appropriate value). The resulting PBM would produce predictions for all components of u. Although the PBM prediction of the third component likely would be quite inaccurate, the CoSTA term could help. Due to the inter-dependencies of the components, this could potentially give a useful prediction of the third component of the displacement, that in turn could make the other components more accurate. This idea is not pursued further in this thesis.

Table 3.5.1: Manufactured solutions for the dimensionally reduced elasticity problems.

Label	$u(t, x, y, \alpha)$
ed1	$\begin{bmatrix} \sin(\pi(x+\alpha y+\frac{1+alpha}{2}z))\cos(\alpha t) \\ \cos(\pi(x+\alpha y+\frac{1+alpha}{2}z))\sin(\alpha t) \\ -\cos(\pi(x+\alpha y+\frac{1+alpha}{2}z))\sin(\alpha t) \end{bmatrix}$
ed2	$\begin{bmatrix} \exp(\frac{(-tx^2+y^2+z^2)}{(1+\alpha+t^2)}) \\ \exp(\frac{(+tx^2-y^2+z^2)}{(1+\alpha+t^2)}) \\ \exp(\frac{(+tx^2+y^2-z^2)}{(1+\alpha+t^2)}) \end{bmatrix}$
ed3	$\begin{bmatrix} x^3+y^2(t+0.5)^{1.5}+xy\alpha+(t+0.5)^{0.5}z^2+z(x+y)\alpha \\ x^2+y^3(t+0.5)^{1.1}-xy\alpha+(t+0.5)^{0.5}z^2+z(x-y)\alpha \\ x^2+y^2(t+0.5)^{1.1}+xy\alpha+(t+0.5)^{0.5}z^3+z(-x+y)\alpha \end{bmatrix}$

Table 3.6.1: Manufactured solutions for nonlinear elastic problems.

Label	$u(t, x, y, \alpha)$	E
n1	$\begin{bmatrix} \sin(\pi(x+\alpha y))\cos(\alpha t) \\ \cos(\pi(x+\alpha y))\sin(\alpha t) \end{bmatrix}$	$\frac{5}{\sqrt{20+\|\|\epsilon\|\|_F}}$
n2	$\begin{bmatrix} \exp(\frac{(-tx^2+y^2)}{(1+\alpha+t^2)}) \\ \exp(\frac{(+tx^2-y^2)}{(1+\alpha+t^2)}) \end{bmatrix}$	$\frac{5}{\sqrt{20+\|\|\epsilon\|\|_F}}$
n3	$\begin{bmatrix} x^3+y^2(t+0.5)^{1.5}+xy\alpha \\ x^2+y^3(t+0.5)^{1.1}+xy\alpha \end{bmatrix}$	$\frac{5}{\sqrt{20+\|\|\epsilon\|\|_F}}$

In chapter 2.3 the linear elasticity equation was written

$$\bar{\boldsymbol{\sigma}} = C\bar{\boldsymbol{\epsilon}}(\boldsymbol{u}) \tag{3.6.1}$$

$$\nabla \cdot \sigma - \ddot{\boldsymbol{u}} = -\boldsymbol{f}, \tag{3.6.2}$$

where C was a matrix. Equation 3.6.1 describes a linear relation between strain and stress. This is a good approximation only for some materials, or for very small strains and stresses. To make a more realistic, nonlinear equation, we replace the assumption that the unit Young's modulus $E = 1$, with a non-constant $E = E(\bar{\boldsymbol{\epsilon}})$, and rewrite the Equation 3.6.1 to

$$\bar{\boldsymbol{\sigma}} = E(\bar{\boldsymbol{\epsilon}})C\bar{\boldsymbol{\epsilon}}(\boldsymbol{u}) \tag{3.6.3}$$

The stiffness of a material usually decreases with applied strain, so we choose a decreasing function

$$E = \frac{5}{\sqrt{20 + ||\epsilon||_F}}, \tag{3.6.4}$$

where $||\cdot||_F$ is the Frobenius norm. This choice ensures that the approximation $E \approx 1$, that is assumed in the linear PBM, holds for values of solutions used in the experiments.

The manufactured solutions used in this experiment are the same as in Section 3.4, and are presented in Table 3.6.1. Due to long computation time, only 500 time steps, 10×10 elements and 5 initializations of DDM and CoSTA were used in this experiment. The DNN learning rate was also increased to $8e-5$.

Chapter 4
Results and Discussion

4.1 Heat with Reduced Dimensionality

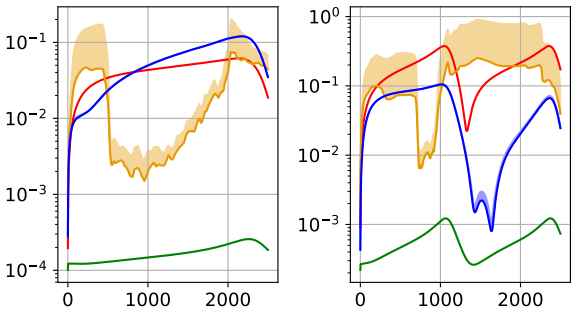
4.1.1 1D model for 2D system

In this section we will present results of the error obtained using the 4 methods tested on the first setup - one-dimensional PBM, DDM and CoSTA, and two-dimensional PBM. We will denote the PBMs 1DPBM and 2DPBM respectfully. We expect 2DPBM to be considerably more accurate than the 1DPBM, since the latter is a rather unsophisticated simplification of the former. Hoping CoSTA can correct for the simplification, the ideal outcome would be if CoSTA is (at least) as accurate as 2DPBM. That being said, the computational cost of predicting with the one-dimensional CoSTA is comparable that of the 1DPBM, while 2DPBM is much more expensive[1]. Thus, a substantial improvement over the one-dimensional PBM should still be considered a victory for CoSTA.

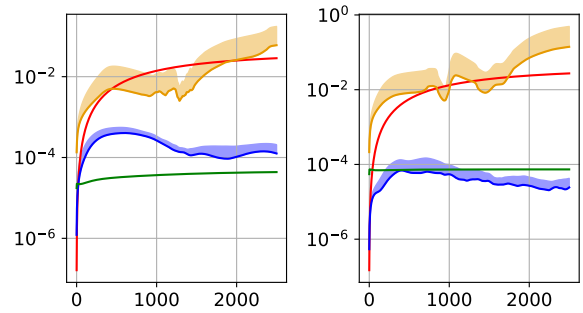
Plots of the errors of the interpolation cases can be found in Figure 4.1.1. As expected, the error from 2DPBM is several orders of magnitude lower than that of 1DPBM, in all cases. For solution *d1*, CoSTA does not seem to improve much over 1DPBM. Although the error of CoSTA is considerably lower than that of 1DPBM for $\alpha = 1.5$, especially for $t > 0.5$, it is worse than 1DPBM for $\alpha = 0.7$, and such an inconsistency in performance countervail the slight overall accuracy boost. For the other solutions, however, CoSTA is comparable to 2DPBM, with an error several orders of magnitude lower than 1DPBM. For solution *d4* it is even more accurate than 2DPBM in both cases. DDM accuracy is comparable to 1DPBM, and outperformed by CoSTA in all cases except intermediate steps in solution *d1*.

Figure 4.1.2 shows the error plots for extrapolation cases. As expected, both CoSTA an DDM have harder times competing with the PBMs in these cases, since only the data dependent methods are affected by the effect of extrapolation. The accuracy of CoSTA compared to the PBM methods vary much more than for interpolation. For solution *d1*, CoSTA is now much worse than 1DPBM in one case, and very similar in the other. For solution *d2*, CoSTA is no longer comparable to 2DPBM in the extrapolation. With $\alpha = -0.5$ it ends up with similar error as 1DPBM only after being less accurate for most of the interval, while for $\alpha = 2.5$ it is much more accurate, yet not very close to 2DPBM. Solution *d3* sees a huge difference between positive and negative α. For $\alpha = -0.5$, the error of the initially accurate CoSTA steadily grows during the modelled time period, and ends up much larger than the 1DPBM error. For $\alpha = 2.5$, on the other hand, CoSTA is about as accurate as 2DPBM, and several orders of magnitude more accurate than 1DPBM. This is a good example of the different nature of the extrapolation cases. While $\alpha = 2.5$ only introduces a more extreme scenario compared to those in the training set, $\alpha = -0.5$ completely reverses the sign of the first term in solution *d3*, which the DDM term in CoSTA evidently is not able to compensate for with the training data used. Only in solution *d4* is CoSTA consistently more accurate than 1DPBM for all values of α. For $\alpha = 2.5$ it is even still more accurate than 2DPBM. DDM accuracy in extrapolation cases varies, with some cases being much worse than 1DPBM, and most cases about equally

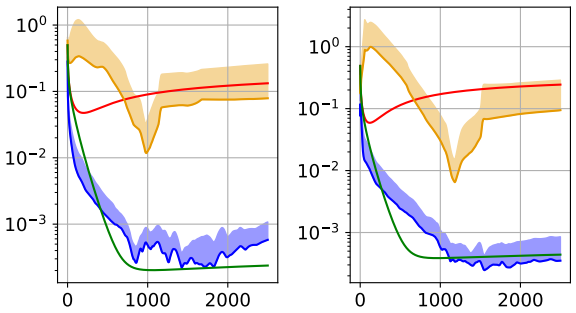
[1] Since the PBM part of a CoSTA is solved twice, in addition to a DDM prediction, the computational cost of CoSTA is slightly more than twice that of the 1DPBM. This is still much less than that of the 2DPBM, which has 20 times as many elements.



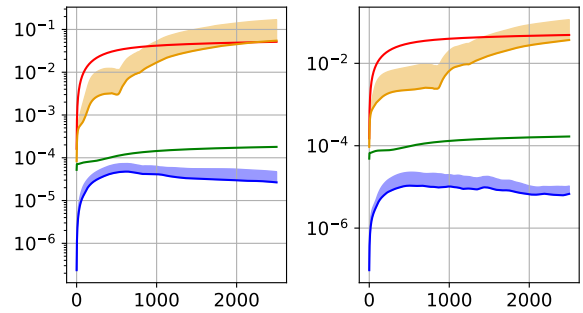
(a) Solution *d1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



(b) Solution *d2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

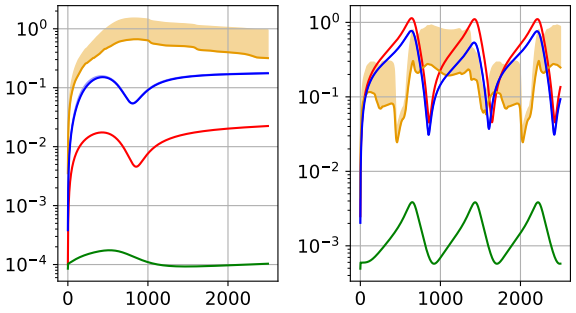


(c) Solution *d3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

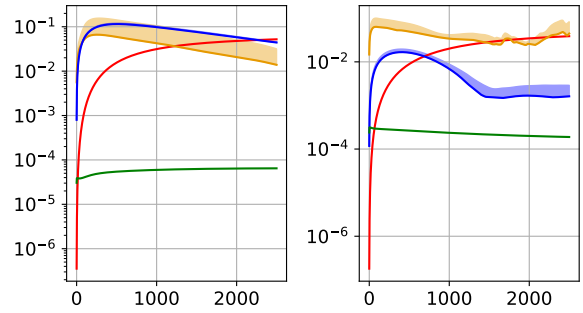


(d) Solution *d4* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

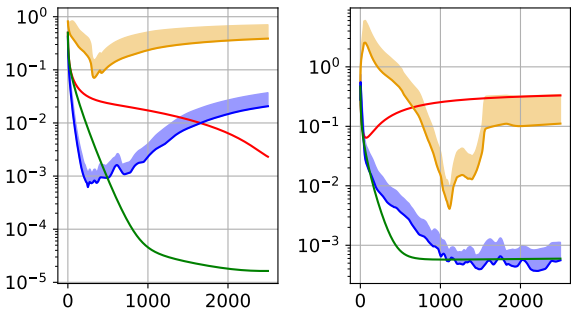
Figure 4.1.1: Temporal development of relative l_2 error for solutions with the first dimensional reduction setup in interpolation scenarios. For solution *d1*, CoSTA is not in general more accurate than the one-dimensional PBM, while for the other solutions, CoSTA is close in accuracy to the two-dimensional PBM. (— 1D PBM, — 2D PBM, — DDM, — CoSTA)



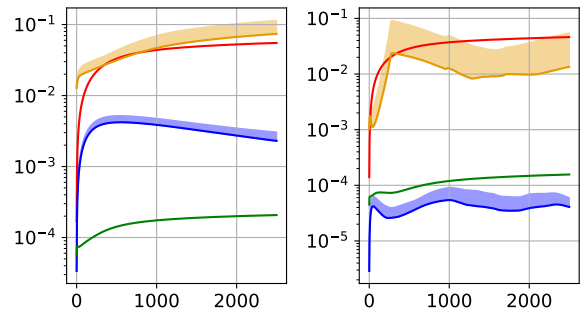
(a) Solution *d1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(b) Solution *d2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(c) Solution *d3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(d) Solution *d4* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

Figure 4.1.2: Temporal development of relative l_2 error for solutions with the first dimensional reduction setup in extrapolation scenarios. The accuracy of CoSTA varies a lot between the cases, from worse than the one-dimensional PBM to better than the two-dimensional. (— 1D PBM, — 2D PBM, — DDM, — CoSTA)

accurate to 1DPBM. Notably, in solutions *d2* and *d4*, the relative accuracy of DDM compared to 1DPBM is slightly better in extrapolation cases than in interpolation cases. The opposite is the case for CoSTA.

Figure 4.1.3 shows the final solutions and predictions in interpolation cases. Here we can see 2DPBM produce good predictions in all cases. For CoSTA, we can see a slight mismatch with the exact solution for solution *d1*, while the others are correct. 1DPBM and DDM are visibly incorrect in all cases, the latter with a very large uncertainty in many cases. Corresponding plots for extrapolation are found in Figure 4.1.4. 2DPBM is still correct in all cases, while CoSTA is visibly incorrect for solution *d1*, and solutions *d2* and *d3* for $\alpha = -0.5$. PBM gives a good prediction for $\alpha = -0.5$ in solution *d3*, and quite bad ones in the other cases. DDM predictions have somewhat correct shape in some cases, but has a large variance in all cases.

Summarizing the results from this section, we see that CoSTA is able to correct for a missing dimension in three of four solutions tested in interpolation scenarios. In extrapolation scenarios the results vary much more, and CoSTA only significantly outperform 1DPBM on half of the cases, while even being considerably worse than 1DPBM in two of the cases with negative α. DDM have a few somewhat decent predictions, but the uncertainty is so large that they cannot be trusted to be correct.

4.1.2 2D model for 3D system

Plots of the errors can be found in Figures 4.1.5 for interpolation cases. We see that the CoSTA framework generally provides predictions several orders of magnitude more accurate than PBM in all cases. It is also more accurate than DDM in all cases, although not by much in solution *d3*. DDM accuracy varies quite a bit for solution *d1*, but is generally worse than PBM. In the other solutions, DDM is much more accurate than PBM. Extrapolation plots are found in Figure 4.1.6. CoSTA is by far most accurate in every case except solution *d2* for $\alpha = -0.5$, where PBM is slightly more accurate, and solution *d3* for $\alpha = -0.5$, where PBM is quite a bit more accurate. DDM is quite heavily affected by the extrapolation. It has a high accuracy for solutions *d1* and *d4* with $\alpha = 2.5$, but has otherwise much larger error, and goes from being more accurate than PBM in most cases, to less accurate than PBM in most cases. CoSTA as well has generally larger error in extrapolation, but the gap between CoSTA and DDM is generally larger in extrapolation than in interpolation, showing that CoSTA was less affected by the extrapolation.

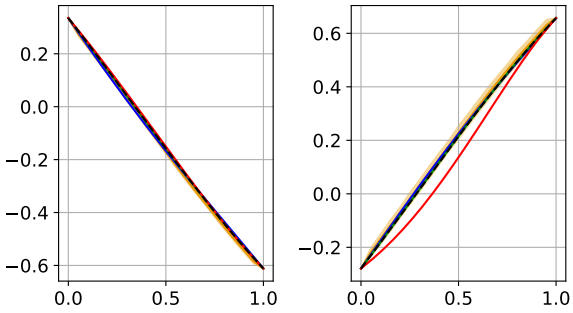
Even when ignoring the 2DPBM in:x the first setup, the results from the second setup are much more promising for CoSTA. This might simply be due to the choices of solutions, as the method is in principle the same independent of the amount of dimensions.

4.2 Nonlinear Heat

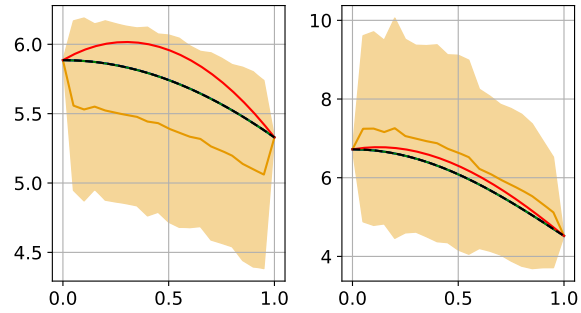
4.2.1 Setup with Slight Nonlinearity

Figure 4.2.1 shows the error plots for interpolation cases. We observe that CoSTA is much more accurate than both of the other two methods for all solutions. DNN accuracy varies around PBM accuracy, but is more often worse than better.

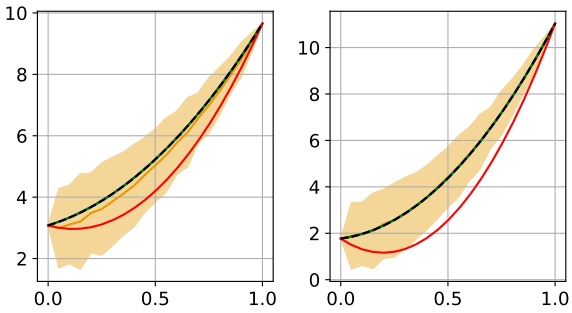
Error plots from the extrapolation are shown in Figure 4.2.2. CoSTA is less accurate than PBM for solutions *c1* and *c2* for $\alpha = -0.5$, and about equal to PBM for solution *c1* with $\alpha = 2.5$. In the 5 other cases, CoSTA significantly outperforms PBM. DDM is worse than PBM



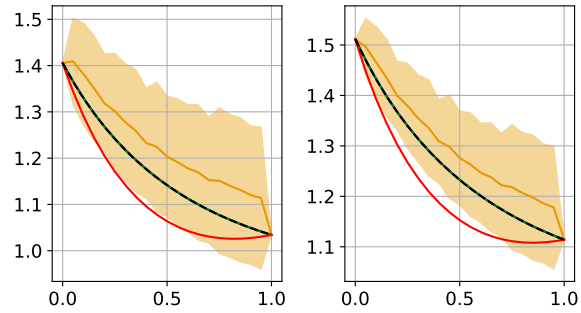
(a) Solution *d1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



(b) Solution *d2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

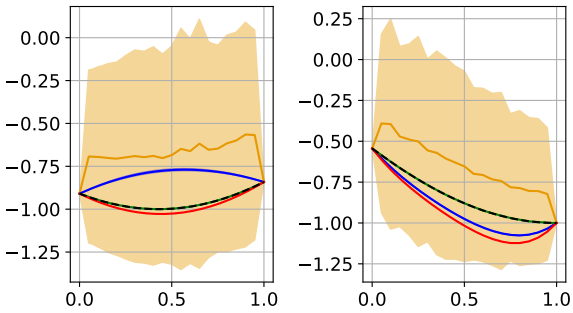


(c) Solution *d3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

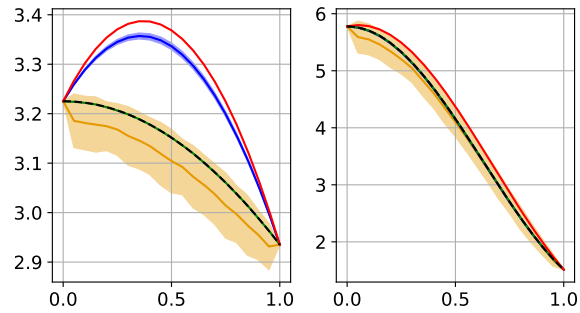


(d) Solution *d4* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

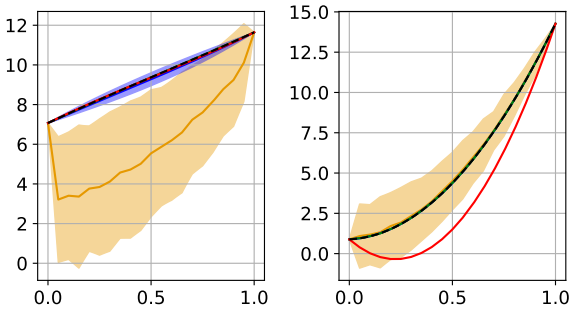
Figure 4.1.3: Plots showing the exact final solution, along with the predictions, for the interpolation cases of the first dimensional reduction setup. 2DPBM is qualitatively correct in all cases. We barely see CoSTA predicting a bit wrong on solution *d1*. 1DPBM is usually a bit off, and DDM predictions have a very high variance. (- - - Exact, — 1D PBM, — 2D PBM, — DDM, — CoSTA)



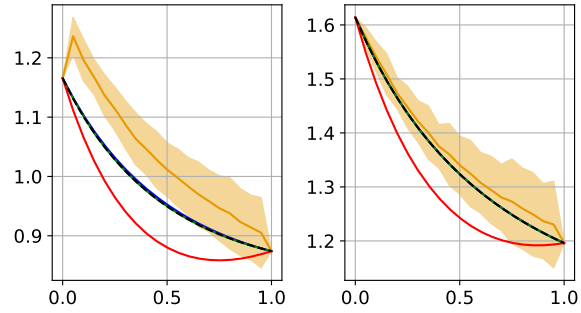
(a) Solution *d1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(b) Solution *d2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

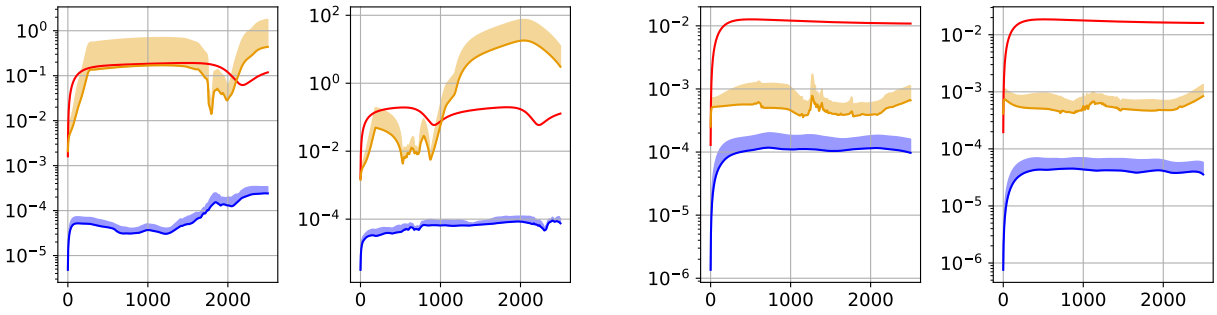


(c) Solution *d3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

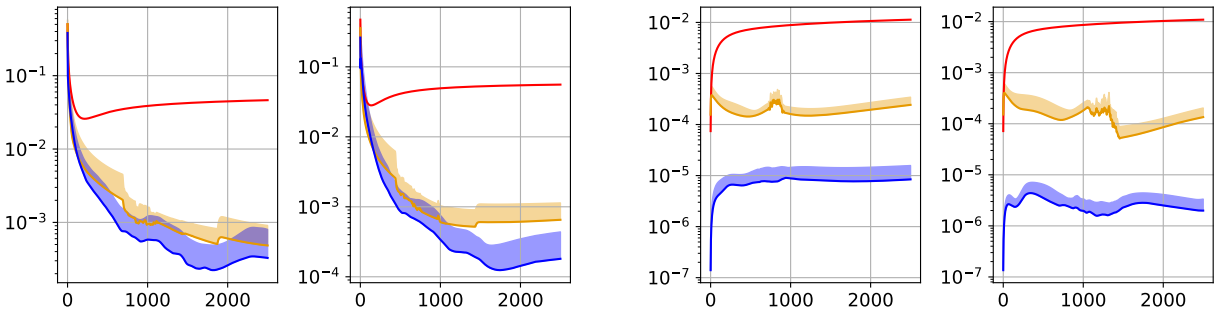


(d) Solution *d4* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

Figure 4.1.4: Plots showing the exact final solution, along with the predictions, for the extrapolation cases of the first dimensional reduction setup. CoSTA give quite bad predictions in four of the plots, but is seemingly spot on in the others. 1DPBM is wrong in most cases, and DDM is again very uncertain. (- - - Exact, — 1D PBM, — 2D PBM, — DDM, — CoSTA)

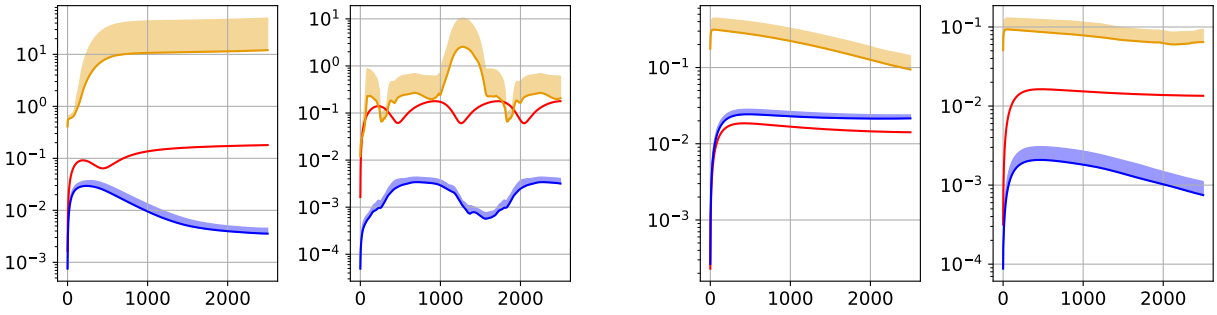


(a) Solution *d1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right) **(b)** Solution *d2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

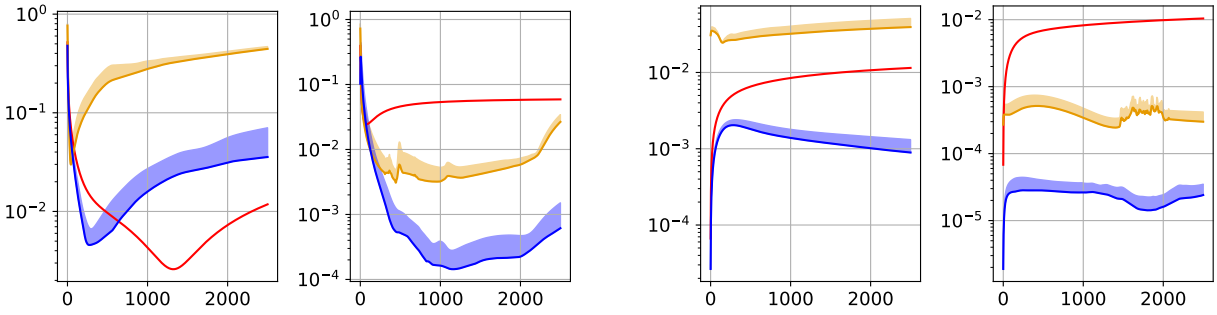


(c) Solution *d3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right) **(d)** Solution *d4* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

Figure 4.1.5: Temporal development of relative l_2 error for solutions with the second dimensional reduction setup in interpolation scenarios. CoSTA error is more accurate than both PBM and DDM for all solutions, often by several orders of magnitude. (— PBM, DDM, CoSTA)



(a) Solution *d1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right) **(b)** Solution *d2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(c) Solution *d3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right) **(d)** Solution *d4* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

Figure 4.1.6: Temporal development of relative l_2 error for solutions with the second dimensional reduction setup in extrapolation scenarios. For $\alpha = -0.5$ on solutions *d2* and *d3*, PBM is more accurate than CoSTA. In all other cases, CoSTA is much more accurate than both of the other methods. (— PBM, DDM, CoSTA)

in most cases, although for solution *c2* with $\alpha = -0.5$, it is more accurate than both the other methods.

The final solutions are plotted in Figures 4.2.3 and 4.2.4 for interpolation and extrapolation respectively. In these figures we see if the methods have gotten the correct shape and size of the solution. We observe that DDM often has an uncertainty so large that there is no value in its prediction. CoSTA predictions are correct in all interpolation tests, whereas PBM are a bit off on some of them. The two extrapolation cases where CoSTA error is bigger than PBM error ($\alpha = -0.5$ for solution *c1* and *c2*), we here see that the correction term has influenced the CoSTA prediction in the wrong direction, as the PBM prediction lies between it and the correct solution.

4.2.2 Setup with More Nonlinearity

Error plots for the experiments with high nonlinearity are presented in Figure 4.2.5 for interpolation and 4.2.6 for extrapolation. The plots are quite similar to the ones with low nonlinearity. The biggest difference is the divergence of DNN predictions of solution *xc1* with $\alpha = 0.7$ and $\alpha = 2.5$. This is discussed in Chapter 4.6.1, since as is argued there, it is not a phenomenon specific to this experiment. We therefore disregard the divergence for now.

For CoSTA and PBM, the difference from the previous experiments are that the error is slightly larger in all cases. The difference between these two methods on the logarithmic scale is very similar in both experiments, indicating that CoSTA works well for linearized PBMs independent of the degree of nonlinearity of the system. The only expection is solution *xc1* for $\alpha = 2.5$, where CoSTA is now worse than PBM, compared to equally accurate with lower nonlinearity.

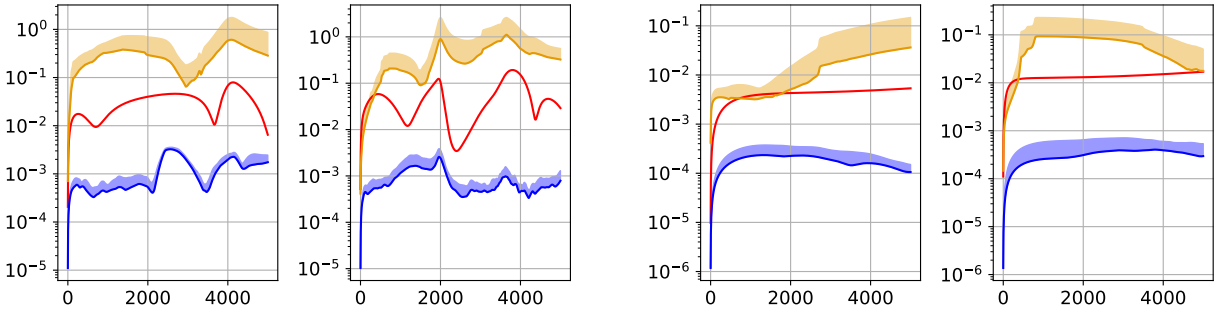
For DDM, both experiments have the same input and correct output, so the error development should only differ due to the randomness of the method. In most cases the difference is relatively minor, and in many cases DNN accuracy relative to PBM is better with higher nonlinearity, as should be expected.

Plots of the final solutions are shown in Figure 4.2.7 for interpolation. As with low nonlinearity, CoSTA is qualitatively correct in all cases, whereas PBM often gets the shape a bit wring, and DDM is highly unreliable. Figure 4.2.8 shows final solutions for extrapolation. Again, CoSTA is on the wrong side of PBM in a few of the cases, but more correct in most cases. For solution *xc2* with $\alpha = -0.5$, DDM is the only method that got the prediction right.

4.3 Elasticity Equation With and Without Known Source Term

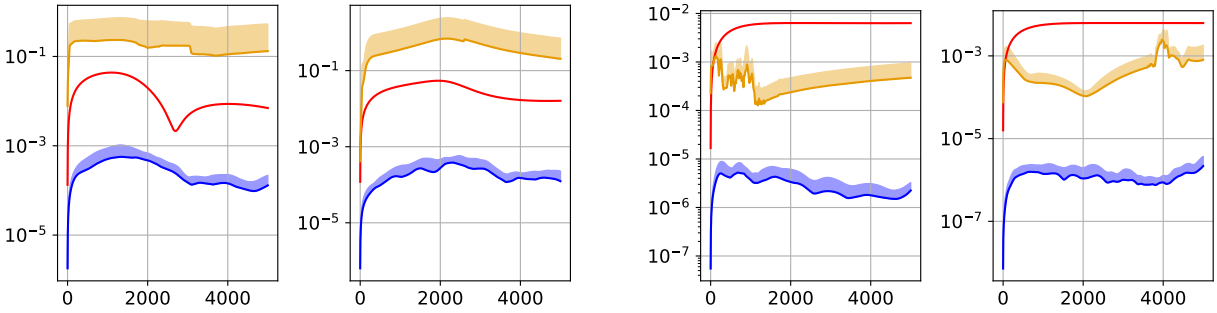
The temporal development of the relative error is presented in Figure 4.3.1 for interpolation scenarios and Figure 4.3.2 for extrapolation scenarios. The results from correct and unknown source terms are shown next to each other for easy comparison. CoSTA gives the most accurate results in all interpolation cases, as well as several extrapolation cases. Only for $\alpha = -0.5$ in solution *e1* with correct source term, and *e2* with zero source term, CoSTA is clearly worse than one of the other methods.

CoSTA consistently outperforms PBM in all of the interpolation scenarios in both tests, showing that even with the correct source term, there is a considerable modelling error in the PBM that the CoSTA is able to correct for. For extrapolation with correct source term, the overall accuracy of these two methods across all solutions are quite similar - sometimes CoSTA is a bit more accurate, sometimes a bit less accurate. Without source term, CoSTA is approximately equally accurate to PBM for solution *e1*, and more accurate than PBM for the other two solutions. DDM is better at interpolation than PBM for solution *e3* using correct



(a) Solution *c1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

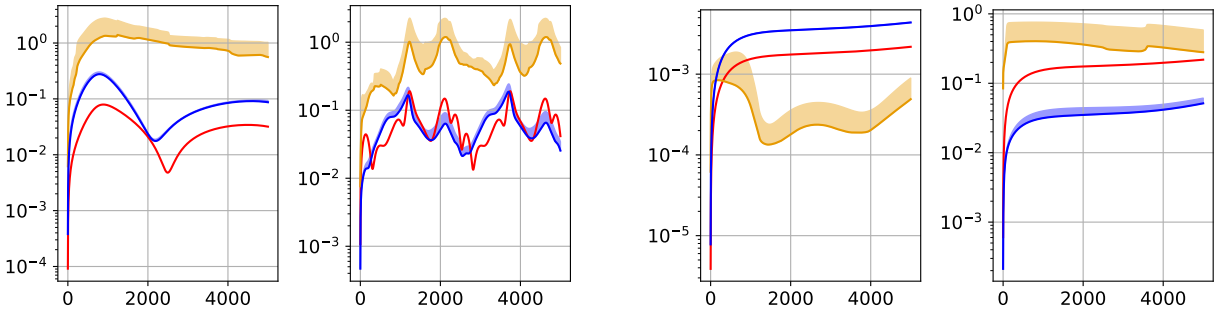
(b) Solution *c2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



(c) Solution *c3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

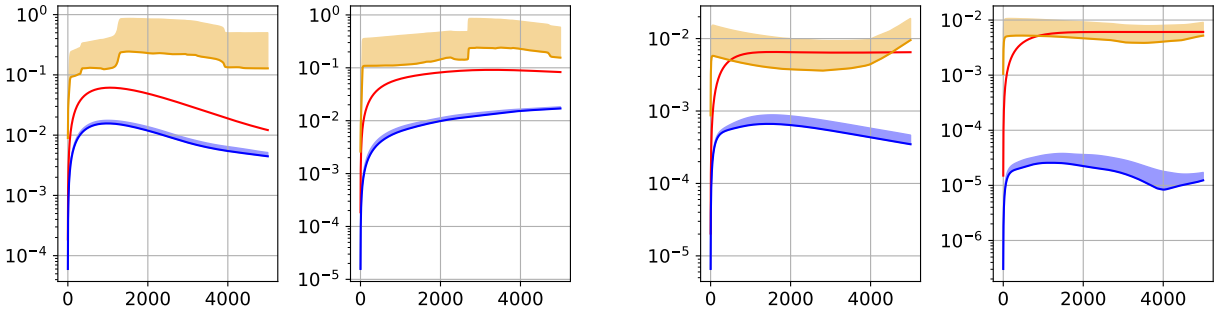
(d) Solution *c4* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

Figure 4.2.1: Temporal development of relative l_2 error for solutions with less temperature dependent conductivity in interpolation scenarios. CoSTA significantly outperforms the other methods in all scenarios. (— PBM, DDM, CoSTA)



(a) Solution *c1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

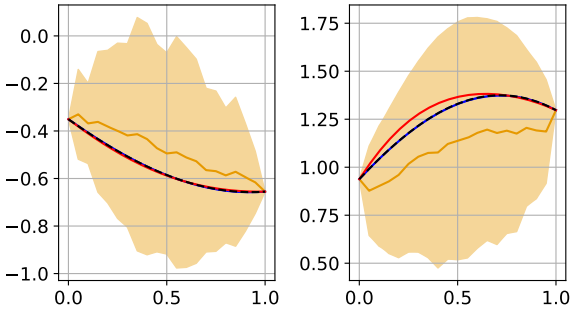
(b) Solution *c2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



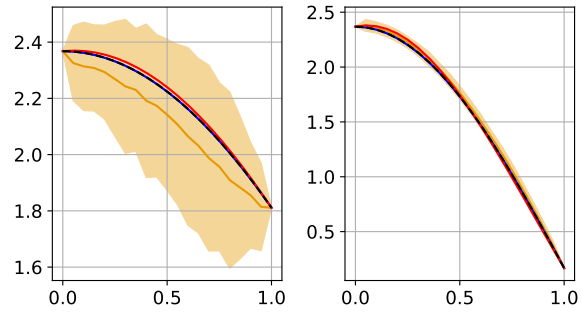
(c) Solution *c3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(d) Solution *c4* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

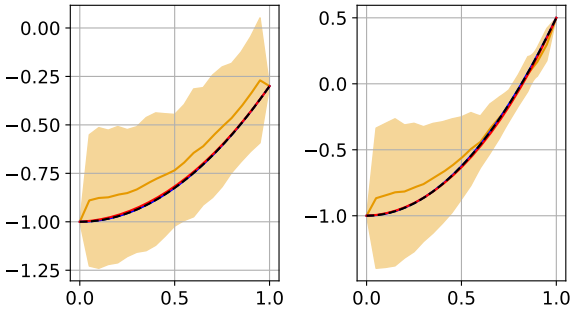
Figure 4.2.2: Temporal development of relative l_2 error for solutions with less temperature dependent conductivity in extrapolation scenarios. CoSTA is more often the most accurate method than not. (— PBM, DDM, CoSTA)



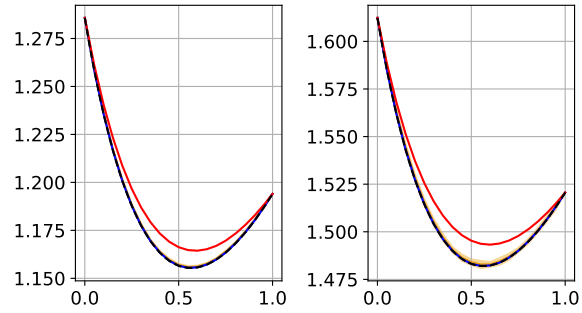
(a) Solution *c1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



(b) Solution *c2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

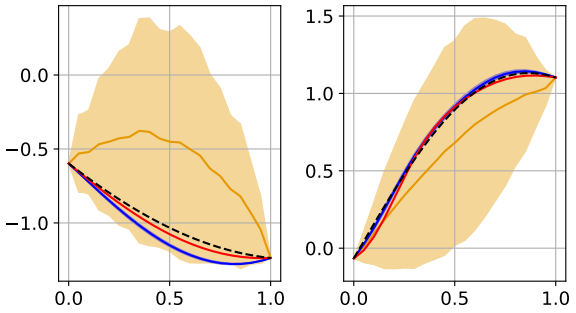


(c) Solution *c3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

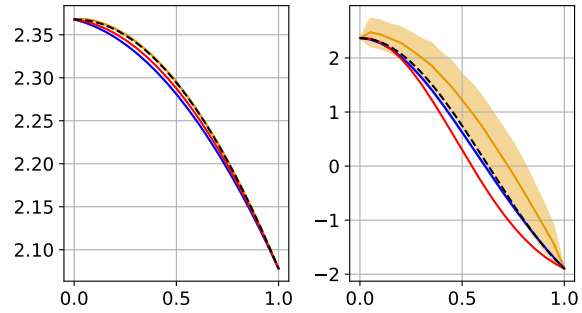


(d) Solution *c4* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

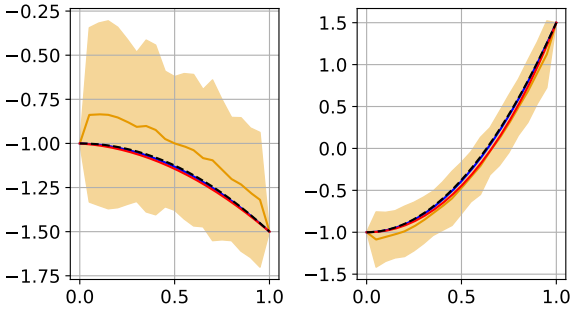
Figure 4.2.3: Final solutions with less temperature dependent conductivity in interpolation scenarios. CoSTA gives precise predictions in all cases, while PBM often is a bit off, and DDM has a very high variance. (- - -Exact, —PBM, DDM, CoSTA)



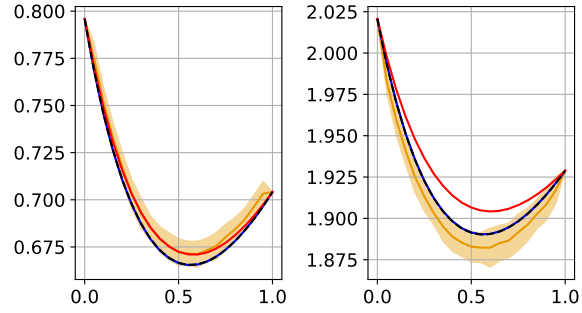
(a) Solution *c1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(b) Solution *c2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

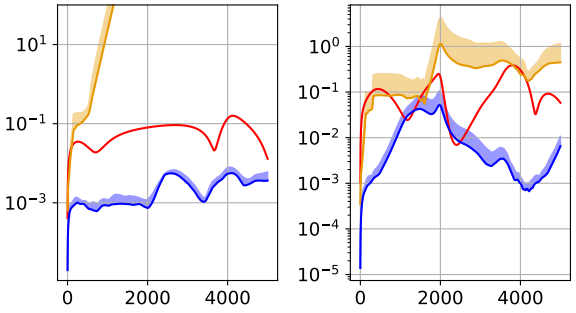


(c) Solution *c3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

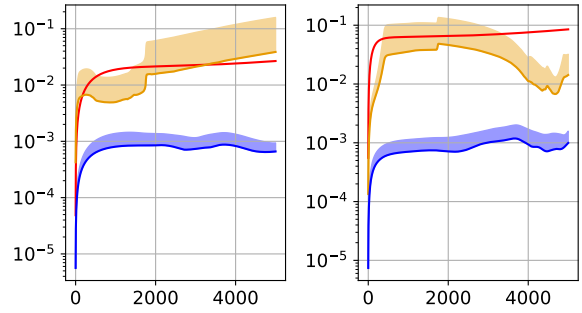


(d) Solution *c4* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

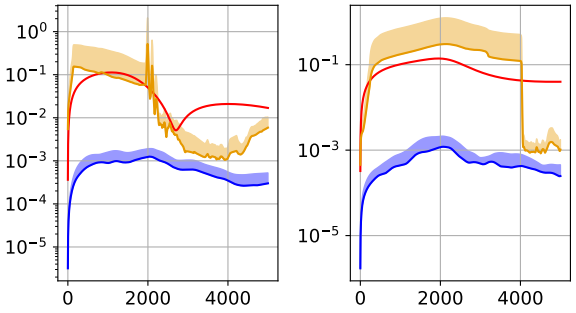
Figure 4.2.4: Final solution with less temperature dependent conductivity in extrapolation scenarios. For solutions *c1* and *c2* with $\alpha = -0.5$, the CoSTA correction influence the prediction in the wrong direction. (- - -Exact, —PBM, DDM, CoSTA)



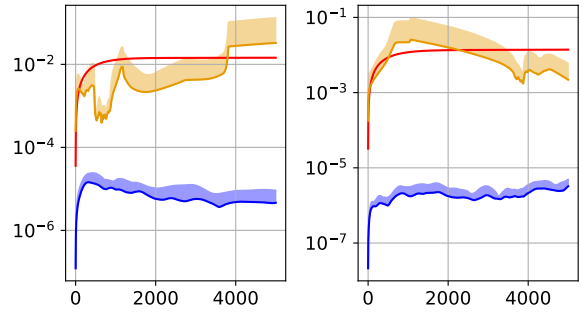
(a) Solution *xc1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



(b) Solution *xc2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

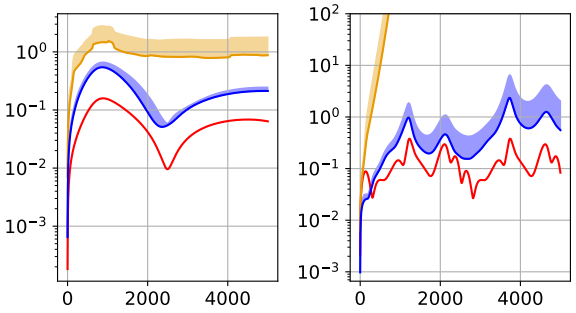


(c) Solution *xc3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

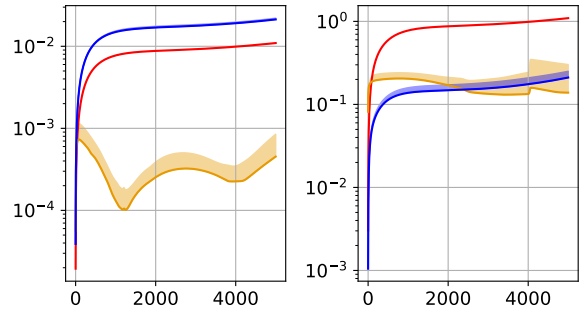


(d) Solution *xc4* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

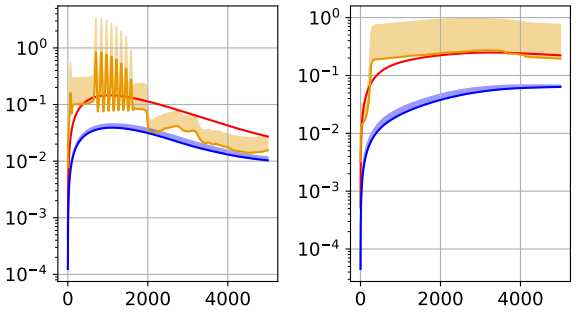
Figure 4.2.5: Temporal development of relative l_2 error for solutions with more temperature dependent conductivity in interpolation scenarios. The results are quite similar to the ones with low nonlinearity. (— PBM, DDM, CoSTA)



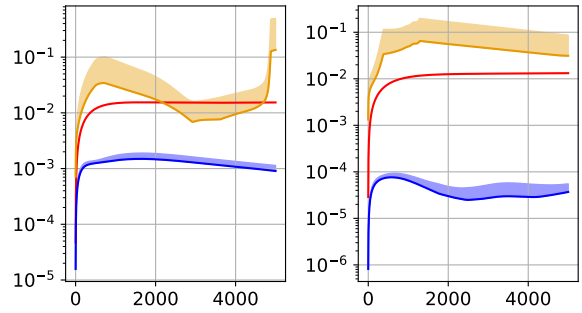
(a) Solution *xc1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(b) Solution *xc2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

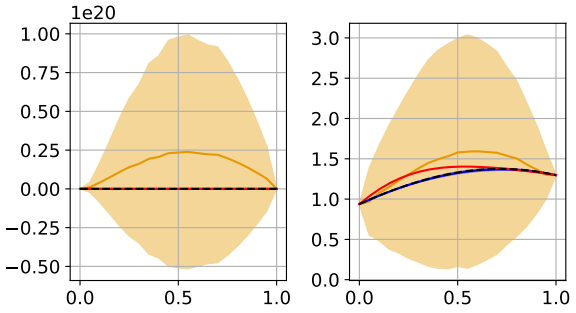


(c) Solution *xc3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

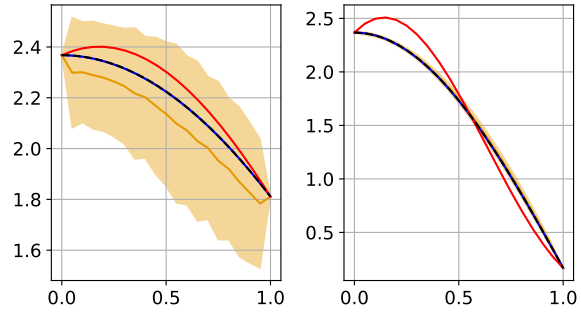


(d) Solution *xc4* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

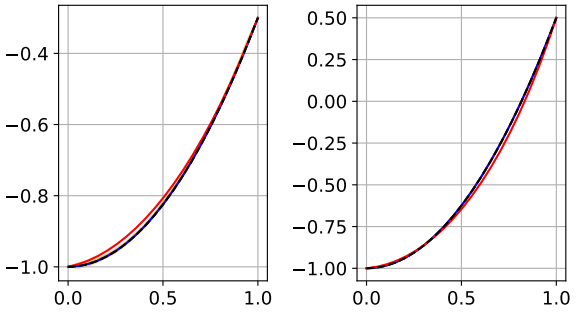
Figure 4.2.6: Temporal development of relative l_2 error for solutions with more temperature dependent conductivity in extrapolation scenarios. The results are quite similar to the ones with low nonlinearity. (— PBM, DDM, CoSTA)



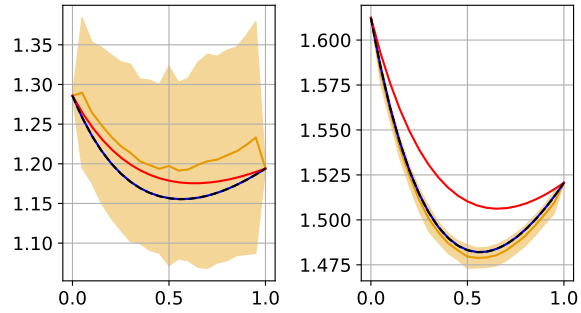
(a) Solution *xc1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



(b) Solution *xc2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

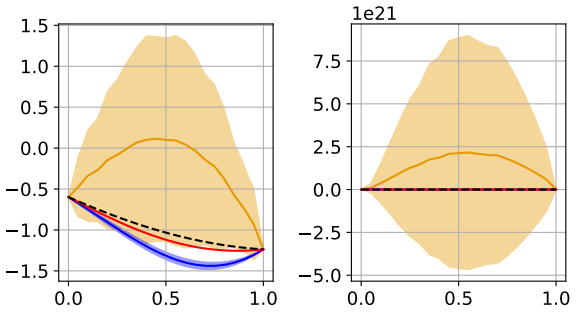


(c) Solution *xc3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

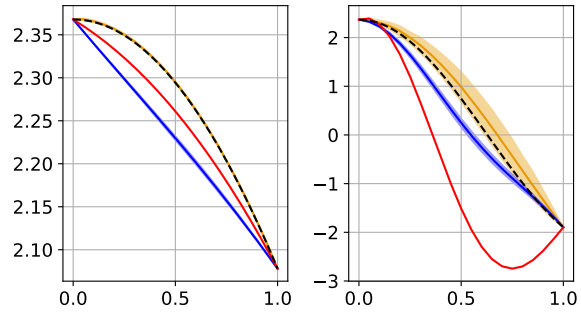


(d) Solution *xc4* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

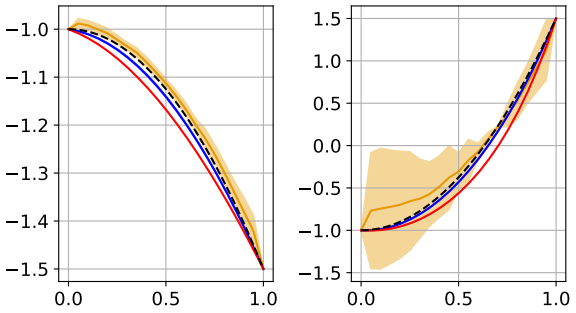
Figure 4.2.7: Final solutions with more temperature dependent conductivity in interpolation scenarios. CoSTA predicts qualitatively correct, while PBM often misses slightly. DDM for solution *xc1* with $\alpha = 0.7$ has diverged. (- - -Exact, —PBM, —DDM, —CoSTA)



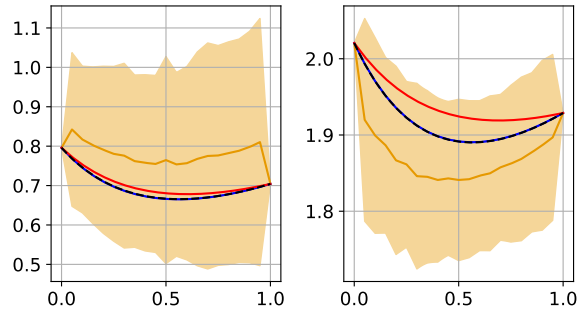
(a) Solution *xc1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(b) Solution *xc2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

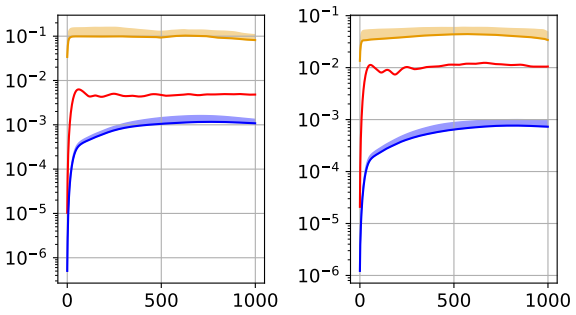


(c) Solution *xc3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

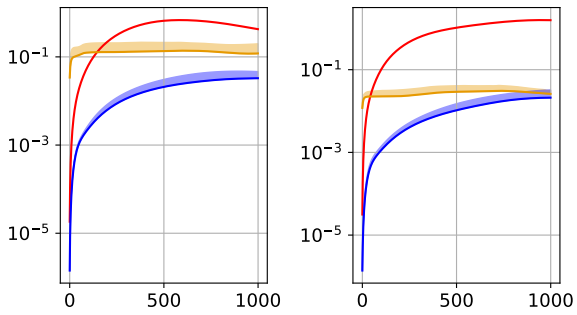


(d) Solution *xc4* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

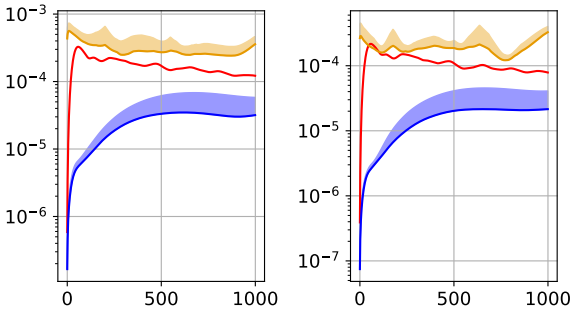
Figure 4.2.8: Final solution with more temperature dependent conductivity in extrapolation scenarios. Results vary from solution to solution. DDM for solution *xc1* with $\alpha = 2.5$ has diverged. (- - -Exact, —PBM, —DDM, —CoSTA)



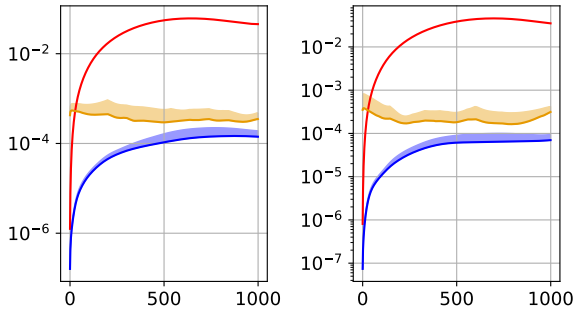
(a) Solution *e1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



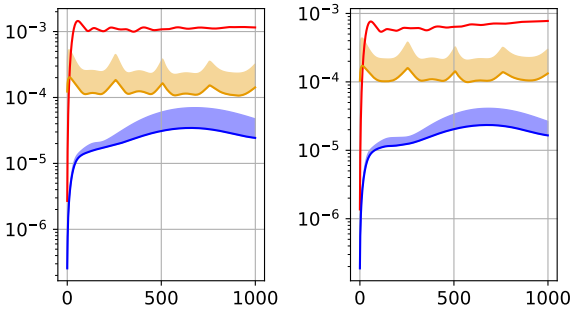
(b) Solution *e1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



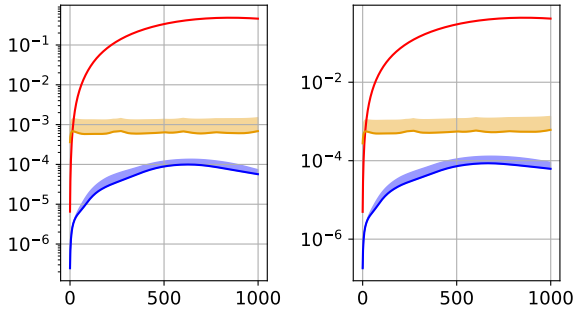
(c) Solution *e2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



(d) Solution *e2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



(e) Solution *e3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)



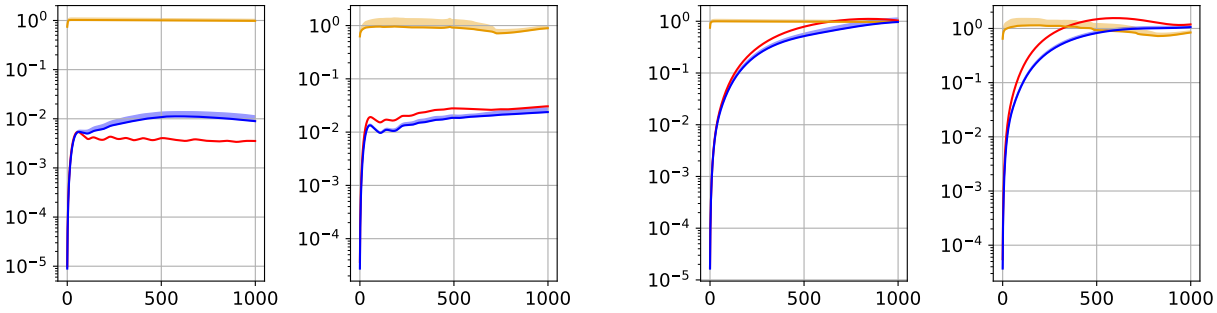
(f) Solution *e3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right)

Figure 4.3.1: Temporal development of relative l_2 error for solutions with correct source term (left) and zero source term (right) in interpolation scenarios. CoSTA is the most accurate method in all the cases. (— PBM, DDM, CoSTA)

source term, and worse in the two other. With unknown source term, PBM is much less accurate than DDM for all solution in the interpolation cases. In the extrapolation cases DDM is much worse than PBM for all solutions with correct source term. With unknown source term, PBM is as accurate as DDM for solution *e1*, and much worse than DDM for the other solutions.

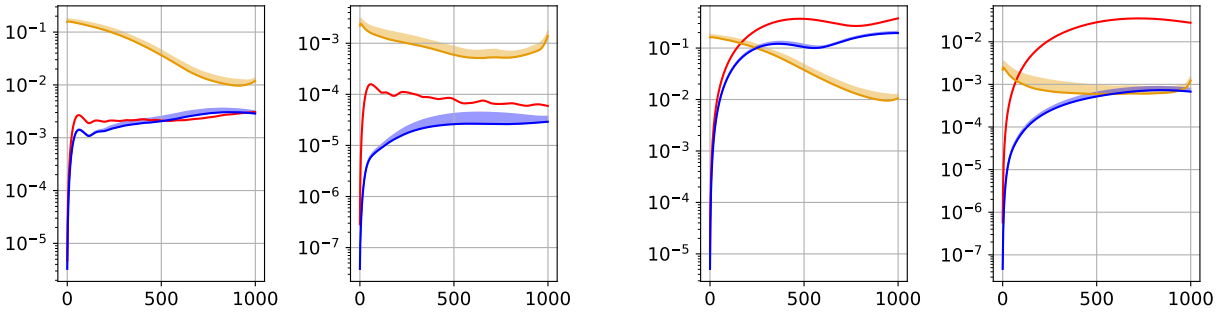
By comparing the results the test with zero source term to the results with correct source term, we can evaluate how well the (additional) modelling error in the zero source model is corrected by the source term. We do not expect CoSTA to provide a perfect correction, and therefore it should perform worse in the zero source term tests than in the correct source terms. This holds in all the cases. In the interpolation cases, CoSTA without correct solution is not that much less accurate than with correct solution, whereas in extrapolation scenarios the difference tends to be larger. It is better to use PBM with knowledge of the source term, than CoSTA without this knowledge, in all cases except interpolation on solution *e3*. But when the modelling error is present, DDM and CoSTA both greatly outperform PBM on most cases.

We also observe by the results in this chapter that the error of CoSTA often fluctuates less



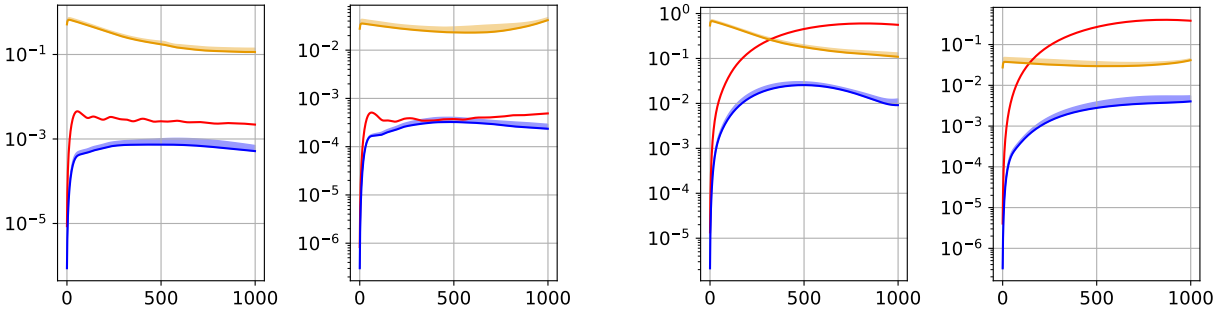
(a) Solution *e1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(b) Solution *e1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(c) Solution *e2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(d) Solution *e2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(e) Solution *e3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

(f) Solution *e3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

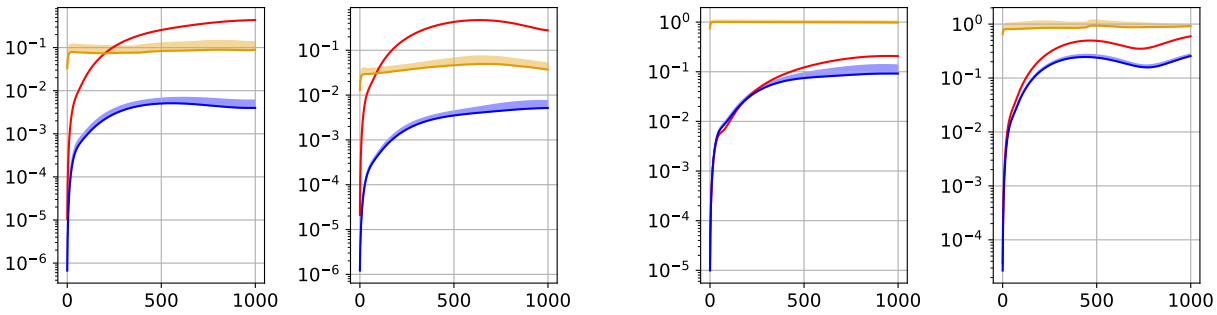
Figure 4.3.2: Temporal development of relative l_2 error for solutions with correct source term (left) and zero source term (right) in extrapolation scenarios. CoSTA is the best method, or among the best methods, in most cases, only being beaten once by each of the other methods. (— PBM, DDM, CoSTA)

than that of DDM. This might indicate more stable predictions.

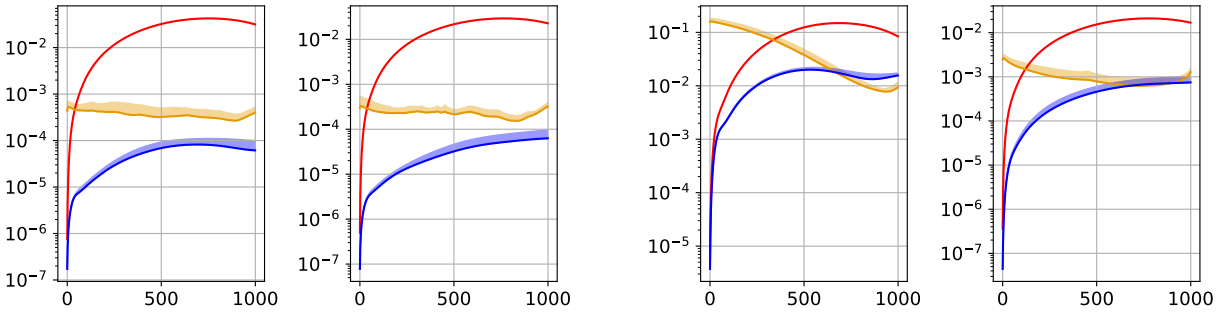
To summarize the results in this chapter, CoSTA is most accurate in all interpolation cases, and most extrapolation cases, independent of whether or not the correct source term is used.

4.4 Linear Elasticity with Reduced Dimensionality

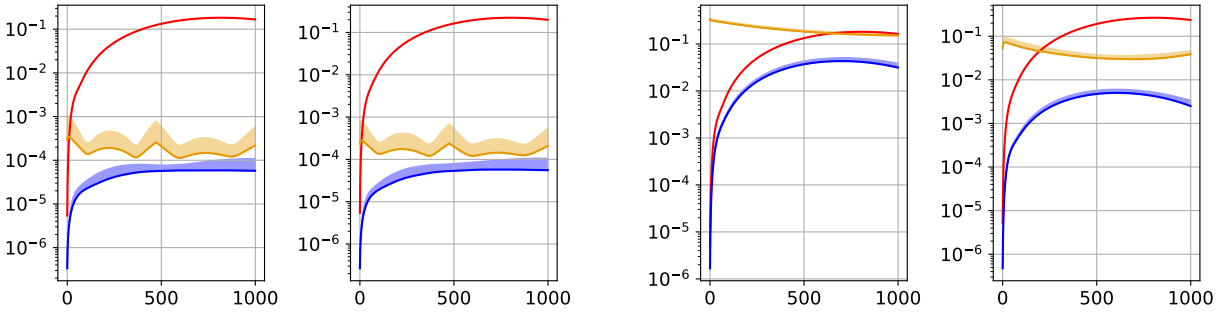
The error development from this experiment is presented in Figure 4.4.1. For interpolation cases, CoSTA is consistently more accurate than both the other two methods. DDM is in turn much better than PBM in these cases. For extrapolation, the DDM and PBM are alternately more accurate than each other, while CoSTA is most accurate except for solution *ed2*, where DDM is most accurate in the last time steps for $\alpha = -0.5$, and about equally accurate to CoSTA for $\alpha = 2.5$



(a) Solution *ed1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right) **(b)** Solution *ed1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(c) Solution *ed2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right) **(d)** Solution *ed2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

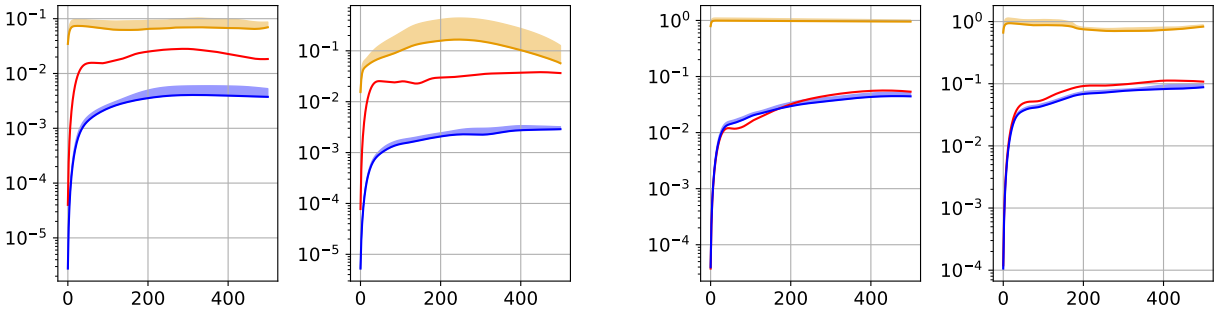


(e) Solution *ed3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right) **(f)** Solution *ed3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

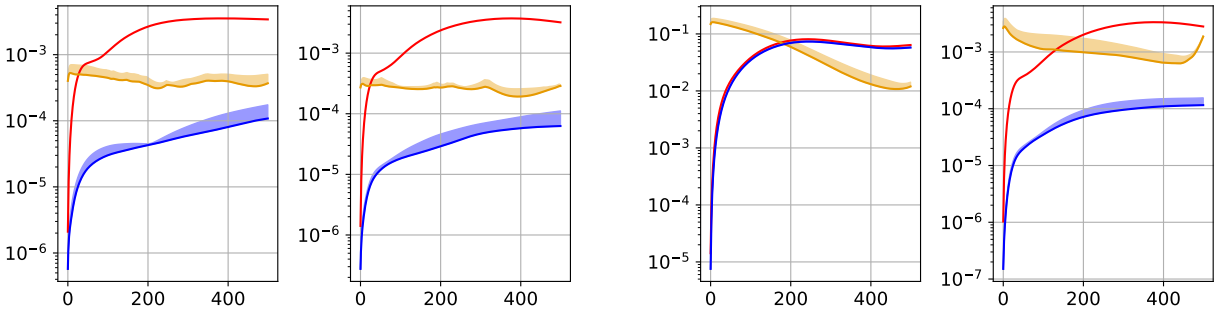
Figure 4.4.1: Temporal development of relative l_2 error for dimensional reduced linear elasticity in interpolation scenarios (left) and extrapolation scenarios (right). We observe that CoSTA is more accurate than PBM in all cases, and better than DDM in all except $\alpha = -0.5$ for solution *ed2*. (— PBM, — DDM, — CoSTA)

4.5 Nonlinear Elasticity

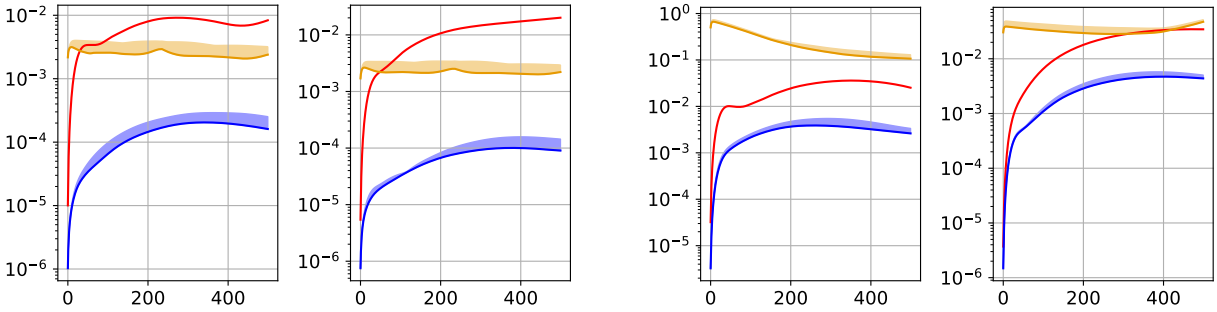
Plots of the error are presented in Figure 4.5.1. For the interpolation, we see that CoSTA much more accurate than PBM and DDM in all the cases. DDM is more accurate than PBM for all interpolation cases except solution *n1*, where PBM is more accurate. In the extrapolation scenarios, PBM and CoSTA are equally accurate in solution *n1* and for $\alpha = -0.5$ in solution *n2*, while CoSTA is more accurate in the other cases. DDM accuracy vary greatly, from being worse than PBM and CoSTA in solution *n1* and solution *n3* with $\alpha = -0.5$, similar to PBM in solution *n3* for $\alpha = 2.5$, between CoSTA and PBM for $\alpha = 2.5$ in solution *n2*, to greatly outperform both the other methods in solution *n2* with $\alpha = -0.5$. This last example is the only case in this experiment where CoSTA is not either significantly better than both the other methods, or tied with the best one of them.



(a) Solution *n1* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right) **(b)** Solution *n1* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(c) Solution *n2* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right) **(d)** Solution *n2* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)



(e) Solution *n3* with $\alpha = 0.7$ (left) and $\alpha = 1.5$ (right) **(f)** Solution *n3* with $\alpha = -0.5$ (left) and $\alpha = 2.5$ (right)

Figure 4.5.1: Temporal development of relative l_2 error for nonlinear elasticity in interpolation scenarios (left) and extrapolation scenarios (right). While CoSTA is most accurate in all interpolation cases, the extrapolation results vary more. (— PBM, DDM, CoSTA)

4.6 Result Comparison and Further Discussion

In this section, we present a summary of the results. With four heat experiments (two with dimensional reduction, two with linearization), each with 4 manufactured solutions, and four elasticity experiments (no modelling errors, zero source term, dimensional reduction and linearization) with 3 manufactured solutions, we have conducted 28 tests for each value of α, a total of 112 tested scenarios. The results have varied a lot both in terms of which methods performed well, and in terms of how large the errors were - some tasks were more difficult than other. In order to summarize the results, we have counted the number of times each of the methods used was the most accurate method, measured by the (mean) RRMSE at the final time step. Table 4.6.1 shows this result, for each value of α and in total. We see that CoSTA is by far the most accurate method. Only for $\alpha = -0.5$ has CoSTA been not best more than sporadically, but even here it has double the amount of wins as each of the other methods. Since PBM is not trained and effected by extrapolation, it is expected that this method performs better in these situations than interpolation, when comparing to the other

Table 4.6.1: Overview of number times each method was the most accurate at final time step, for each value of α and in total.

α	PBM	DDM	CoSTA
-0.5	7	7	14
0.7	1	0	27
1.5	0	0	28
2.5	1	2	25
Total	9	9	94

methods. But DDM is known to be heavily effected by this. Since CoSTA is a combination of PBM and DDM, where data dependent prediction is only a part of the solution, one might assume that its generalizability should be better than that of DDM. It is therefore a bit odd that CoSTA looses to DDM in several extrapolation cases, but no interpolation cases. A possible explanation might be that CoSTA for some reason simply is not as generalizable as DDM. Another may be that these problems with $\alpha = -0.5$ are so different from the training data, that both methods struggle, and it is more arbitrary than in the interpolation cases which one struggles more. In any case, CoSTA is still by far most accurate in these situations as well.

The method of simply counting number of wins is of course prone to misrepresenting the results, since the magnitude of the difference in error is not considered, only the sign. Also, it does not separate second or third place. Figure 4.6.1 shows how many times each method won with a factor of at least x separating it from the next best method. Similarly, Figure 4.6.2 shows the amount of times each model was the worst, and by what factors. We see that CoSTA clearly outclasses the other methods both by number of losses and wins, by any required level of significance.

The variance of both DDM and CoSTA predictions have varied quite a bit throughout the experiments. Consistency is important for predictability, and for only requiring one or a few models to make a reliable prediction. One way of penalizing the variance when summarizing the results is to compare final mean error plus a standard deviation. The dotted lines in Figures 4.6.1 and 4.6.2 show the results of such a penalizing. We see that CoSTA and DDM both perform a bit worse, and PBM a bit better, with this method of evaluation. But the difference between the dotted and solid lines is quite insignificant. In other words, the difference in accuracy between the methods, are generally much larger than the difference in accuracy between the initialization of one method. [2] We conclude that CoSTA is the indisputable winner of the methods even when considering its worst probable predictions.

Table 4.6.2 shows the number of wins each method had for each PDE, dimensionality and type of modelling error. By model dimensionality we mean of the domain modelled. Keep in mind that the different classifications by no means are independent - for example, all elasticity problems are two-dimensional. The manufactured solutions also play an important role for which experiments yield which results. We notice that PBM perform better on the heat equation than the elasticity equation. For the experiments with no extra modelling error, and zero source (which are elasticity), we see a clear difference in DDM wins. Since the DDM in these experiments are practically the same, these differences are due to better PBM, and thus better CoSTA, when the source term is known.

One of the take-away from most experiments is that results highly depend on the solution.

[2] This is easy to see in the error plots - while the blue shaded area showing the standard deviation of CoSTA error at times is relatively thick, the distance to the other lines is usually considerably larger. Remember also that the thickness of the shaded area is also logarithmicaly scaled, so when the mean prediction is very accurate, a thick line does not indicate a big variance. For example, in Figure 4.5.1e, despite the variance, CoSTA is consistently much more accurate than the other methods.

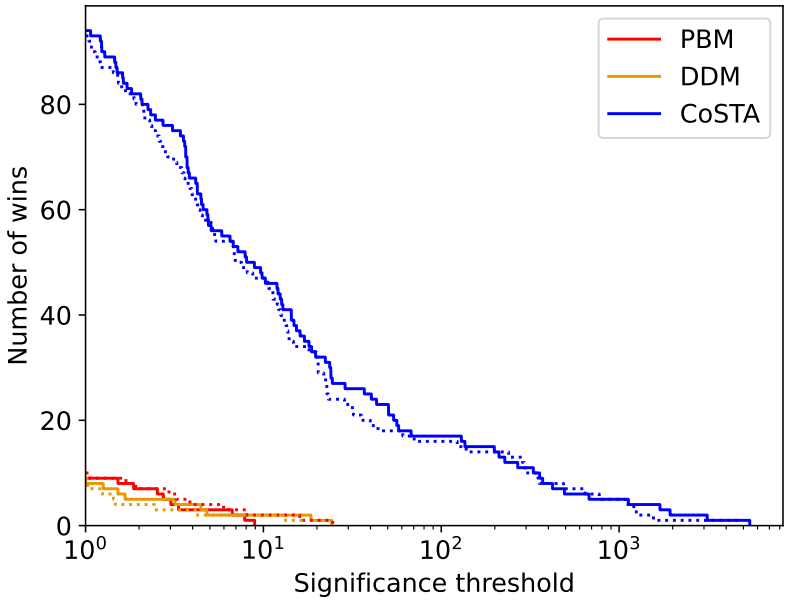


Figure 4.6.1: Overview of number of times each method was the most accurate at the final time step, at various degrees of significance. For a value x_1 on the x-axis, the height of a curve is the number of times the method was better (had lower error) than both the other methods by a factor of at least x_1. E.g. by observing the blue line at 10^2, we see that in close to 20 of the 112 tests, CoSTA won by a factor of at least 10^2 (meaning both PBM and DDM had an error of more than 10^2 times that of CoSTA). The solid line is the results of evaluating the means (of the CoSTA and DDM initializations), while the dotted line is from evaluating the mean + one standard deviation, as a way of penalizing inconsistency.

Elasticity solutions are the same (only extended to 3D in the dimensional reduction chapter), and can be compared across the chapters. In all elasticity chapters, the worst results for CoSTA has been extrapolation of the first solution (*e1*, *ed1* and *n1*). Interestingly, the heat diffusion experiments also often has the worst CoSTA performance in the first two solutions in each experiment. Common to all these solutions is the harmonic functions involved in them. When studying the plots of the final solutions, shown in the either the relevant sections or the Appendix A, these functions tend to vary more along the axes.

Many of manufactured solutions used in the different chapters are similar to each other. Same or similar solutions open for easier comparison across experiments, while differences lets us test a wider range of solutions. The fact that the manufactured solutions vary between the different experiments, may also be a reason for some of the varied performance in the classifications in Table 4.6.2. More importantly, it is also possible that the choice of solutions has contributed to better (or worse) results for CoSTA than it would have in most real world applications. So even though CoSTA has shown remarkable results in this project, we emphasize that it will not necessarily be as good in future use cases. But it should be clear from this thesis that it is worth testing, as it might very well be.

4.6.1 Diverging Predictions

In this section, we discuss the divergence of the DDM prediction in Chapter 4.2.2. Error plots of solution *xc1* with $\alpha = 0.7$ (in Figure 4.2.5) and $\alpha = 2.5$ (in Figure 4.2.6) shows the DDM error exploding at the start of the period. Although not shown in these plots, the mean

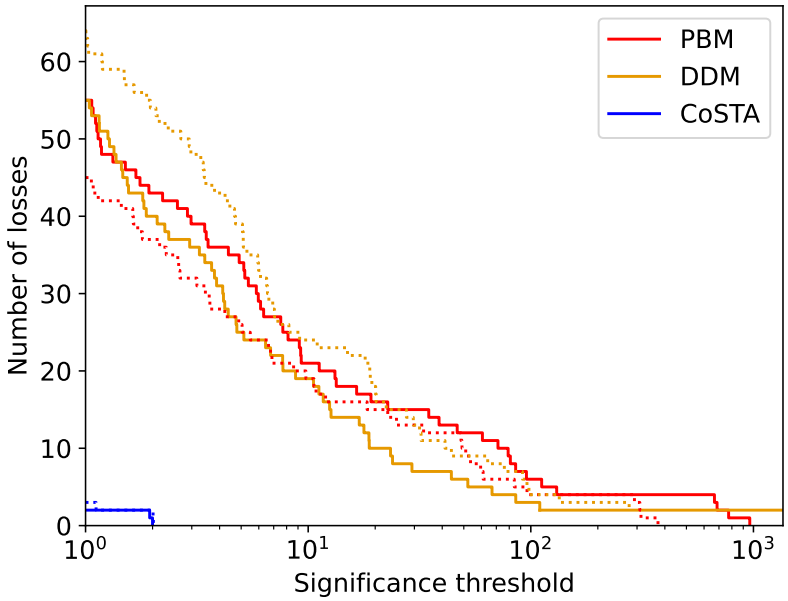


Figure 4.6.2: Overview of number of times each method was the *least* accurate at the final time step, at various degrees of significance. For a value x_1 on the x-axis, the height of a curve is the number of times a method was worse than both the other methods by a factor of at least x_1. The solid line is the results of evaluating the means (of the CoSTA and DDM initializations), while the dotted line is from evaluating the mean + one standard deviation, as a way of penalizing inconsistency. We see that CoSTA only lost two to three times, and never had an error bigger than about twice the error of the second worst method.

Table 4.6.2: Overview of number times each method was the most accurate at final time step, by various classifications.

	PBM	DDM	CoSTA	Total
Equation modelled				
Heat eqn	8	4	52	64
Elastic eqn	1	5	42	48
Model dimensionality				
1D	6	4	38	48
2D	3	5	56	64
Extra modelling error				
None	1	0	11	12
Zero source term	0	3	9	12
Dimensional reduction	5	2	37	44
Linearization	3	4	37	44

RRMSE ends up at around 10^{20} in both the cases, after growing exponentially (linearly on the plots with logarithmically scaled axis) for the whole period. This is due a single model misbehaving, out of the 10 initializations. This is the only example of this included in the thesis, but in the many experiments excluded from the thesis[3], this has happened several

[3]Excluded experiments includes test runs with fewer elements and time steps, not well though through solutions with the modelling error independent of α, or runs with various bugs. The experiments included in the thesis are by no means chosen based on their results.

times, and for a variety of scenarios (often, but not always, with less time steps, and thus less training data). For this reason, this should not be interpreted as result to this one experiment, but as an inherent problem with using DNN in the way it has been used in this thesis. Even though it usually predicts well, it may suddenly diverge without warning. Notice that this only happens for two out of four values of α - for the others, there is seemingly nothing unusual about the results, despite that it is the same instance of the model making the predictions. This unpredictability is a good example of why machine learning have had such a hard time gaining trust in high-stake applications.

When such a divergence first have started, it keeps growing for the entire period. Since the models train exclusively on exact input, the input from previous steps with extreme values are very different from the training input, and hence the network does not work well on these values. Ideally, the networks should have noticed that their predictions no longer correspond to the boundary conditions, for which the input is exact. But again, since the methods have only trained with exact solutions as input, there they have no reason to trust the boundary values more than any other.[4] Because of the sudden but extreme nature of this phenomenon, it is luckily very easy to detect it, and replace the network by training another. Nevertheless, these results should serve as a healthy reminder that black box models may be treacherous, and that proportional safety measures should be taken before applying such models for risky applications.

Sudden divergence similar to this has also happened with CoSTA, although far less frequent. This may be due to the nature of the DNN in CoSTA. These are only trained to adjust the FEM method, which in most cases is relatively close to the exact solution, meaning the output from the DNN in the CoSTA usually lead to relatively small corrections in the solution. The pure DDM on the other hand, has much more influence on the prediction, making it more prone to divergence like this. For the CoSTA method, an alternative and more robust approach for detecting errors than simply checking for extreme values, is to calculate physical properties of the solution that have known or expected ranges of values. This idea is discussed in [44].

[4]There might be ways to prevent this, for example by training the network to focus extra on boundary nodes. Adding noise to the inner nodes of the training input was attempted during this project, but it did not show any significant success, and the idea was abandoned.

Chapter 5
Conclusion and Future Work

5.1 Conclusions

Through numerous experiments, we have in this thesis tested the ability of the corrective source term approach (CoSTA) to outperform its data driven model (DDM) and more or less erroneous physics based model (PBM) counterparts by adding a data driven correction term to the PBM. We have used simplified PBMs to model two- and three-dimensional heat diffusion using reduced dimensionality, and one-dimensional nonlinear heat diffusion using linearization. We have modelled linear elasticity using a PBM without extra modelling errors, with unknown source term, and using dimensional reduction. Finally we have modelled non-linear elasticity using a linear PBM. Throughout the tests, CoSTA has shown an impressive accuracy in the interpolation tests, for which it has been by far the most accurate method in almost all cases. For extrapolation, although the results have been slightly less one-sided, CoSTA is still by far the overall most persistent method in these cases too.
The major conclusions can be summarized as:

- CoSTA has shown a remarkable accuracy for scenarios similar to training data, across all the different experiments
- CoSTA also generalizes well enough to unseen data that it outclasses the other two methods also in these cases.
- Despite some variance in the predictions, CoSTA is consistent enough that its relatively bad predictions generally still by far outperform the other methods.

We can also conclude that the remarkable results that CoSTA has has shown in previous publications generalizes to the type of problems seen in this thesis.

5.2 Future Work

CoSTA has shown promising results, both in this thesis and the previous works [44, 45], but is still not widely tested. As the DDM part of CoSTA is a black box model, the method for verifying its usefulness is through thorough testing. Tests can be done for yet other PDEs, or other modelling applications, with yet other modelling errors, but perhaps more interesting is to test other building blocks of the CoSTA, namely the PBM and DDM. As for now, only the finite difference method and the finite element method has been used as PBM, both of first order (for which they are very similar), and DNN has been used as DDM, with the same architecture each time. If CoSTA is to be used instead of state of the art PBM or DDM models, it should be tested with these as its building blocks. This might involve higher order elements, or other DDMs, depending on the application.

We have seen that the performance of the correction of CoSTA vary a lot from solution to solution. In both this thesis, and the other works written about CoSTA, quite arbitrarily chosen manufactured solutions has been used. Before seeing CoSTA used in the industry, we should have it tested on real world problems, with real world data.

Bibliography

[1] Sondre Sørbø. Physics guided neural network-assisted corrective source term approach to hybrid analysis and modelling. Report for the course TMA4500 - Industrial Mathematics, Specialisation Project, NTNU, 2021. Available online at `https://github.com/sondsorb/CoSTA_PGNN_project_report`.

[2] Han Wei, Shuaishuai Zhao, Qingyuan Rong, and Hua Bao. Predicting the effective thermal conductivities of composite materials and porous media by machine learning methods. *International Journal of Heat and Mass Transfer*, 2018.

[3] Pierre Baldi, Kevin Thomas Bauer, Clara Eng, Peter Sadowski, and Daniel Whiteson. Jet substructure classification in high-energy physics with deep neural networks. *Physical Review D*, 93:094034, 2016.

[4] Gabriel Ibarra-Berastegi, Jon Sáenz, Ganix Esnaola, Agustín Ezcurra, and Alain Ulazia. Short-term forecasting of the wave energy flux: Analogues, random forests, and physics-based models. *Ocean Engineering*, 104:530–539, 2015.

[5] Xiaowei Jia, Jared D. Willard, Anuj Karpatne, Jordan S. Read, Jacob A. Zwart, Michael S. Steinbach, and Vipin Kumar. Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles. *ACM/IMS Transactions on Data Science*, 2:1 – 26, 2021.

[6] Omer San, Adil Rasheed, and Trond Kvamsdal. Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution. *GAMM-Mitteilungen*, 44(2):e202100007, 2021.

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

[8] Jared D. Willard, Xiaowei Jia, Shaoming Xu, Michael S. Steinbach, and Vipin Kumar. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys (CSUR)*, 2022.

[9] Mauricio A. Álvarez, David Luengo, and Neil D. Lawrence. Latent force models. In *AISTATS*, 2009.

[10] Tom Beucler, Stephan Rasp, Michael S. Pritchard, and Pierre Gentine. Achieving conservation of energy in neural network emulators for climate modeling. *ArXiv*, abs/1906.06622, 2019.

[11] Omri Azencot, N. Benjamin Erichson, Vanessa Lin, and Michael W. Mahoney. Forecasting sequential data using consistent koopman autoencoders. *ArXiv*, abs/2003.02236, 2020.

[12] Michael Gauding and Mathis Bode. Using physics-informed enhanced super-resolution generative adversarial networks to reconstruct mixture fraction statistics of turbulent jet flows. *Lecture Notes in Computer Science*, 2021.

[13] Jinlong Wu, Karthik Kashinath, Adrian Albert, Dragos B. Chirila, Prabhat, and Heng Xiao. Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. *J. Comput. Phys.*, 406:109209, 2020.

[14] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations. *CoRR*, abs/1711.10561, 2017.

[15] Maziar Raissi, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, 2019.

[16] Jean-Christophe Loiseau and Steven L. Brunton. Constrained sparse galerkin regression. *Journal of Fluid Mechanics*, 838:42 – 67, 2018.

[17] Yibo Yang and Paris Perdikaris. Physics-informed deep generative models. *ArXiv*, abs/1812.03511, 2018.

[18] Nikhil Muralidhar, Mohammad Raihanul Islam, Manish Marwah, Anuj Karpatne, and Naren Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. *2018 IEEE International Conference on Big Data (Big Data)*, pages 36–45, 2018.

[19] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical review letters*, 98 14:146401, 2007.

[20] J. Nagoor Kani and Ahmed H. Elsheikh. Dr-rnn: A deep residual recurrent neural network for model reduction. *ArXiv*, abs/1709.00939, 2017.

[21] Bin Mu, Bo Qin, Shijin Yuan, and Xiaoyun Qin. A climate downscaling deep learning model considering the multiscale spatial correlations and chaos of meteorological events. *Mathematical Problems in Engineering*, 2020:1–17, 2020.

[22] You Xie, Erik Franz, Mengyu Chu, and Nils Thürey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Trans. Graph.*, 37:95:1–95:15, 2018.

[23] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. *ArXiv*, abs/1806.07366, 2018.

[24] 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.

[25] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6, 2020.

[26] Nick Winovich, Karthik Ramani, and Guang Lin. Convpde-uq: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains. *J. Comput. Phys.*, 394:263–279, 2019.

[27] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[28] Jordan S. Read, Xiaowei Jia, Jared Willard, Alison P. Appling, Jacob A. Zwart, Samantha K. Oliver, Anuj Karpatne, Gretchen J. A. Hansen, Paul C. Hanson, William Watkins, Michael Steinbach, and Vipin Kumar. Process-guided deep learning predictions of lake water temperature. *Water Resources Research*, 55(11):9173–9190, 2019.

[29] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, pages 558–566, 2019.

[30] Junde Lu and Furong Gao. Model migration with inclusive similarity for development of a new process model. *Industrial & Engineering Chemistry Research*, 47:9508–9516, 2008.

[31] Linkai Luo, Yuan Yao, and Furong Gao. Bayesian improved model migration methodology for fast process modeling by incorporating prior information. *Chemical Engineering Science*, 134:23–35, 2015.

[32] Wenjin Yan, Shuangquan Hu, Yanhui Yang, Furong Gao, and Tao Chen. Bayesian migration of gaussian process regression for rapid process modeling and optimization. *Chemical Engineering Journal*, 166:1095–1103, 2011.

[33] U. Forssell and P. Lindskog. Combining semi-physical and neural network modeling: An example ofits usefulness. *IFAC Proceedings Volumes*, 30(11):767–770, 1997. IFAC Symposium on System Identification (SYSID'97), Kitakyushu, Fukuoka, Japan, 8-11 July 1997.

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

[35] Omer San and Romit Maulik. Machine learning closures for model order reduction of thermal fluids. *Applied Mathematical Modelling*, 60:681–710, 2018.

[36] Omer San and Romit Maulik. Neural network closures for nonlinear model order reduction. *Advances in Computational Mathematics*, 44:1717–1750, 2018.

[37] Anuj Karpatne, William Watkins, Jordan S. Read, and Vipin Kumar. Physics-guided neural networks (PGNN): an application in lake temperature modeling. *CoRR*, abs/1710.11431, 2017.

[38] Eric J. Parish and Karthik Duraisamy. A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305:758–774, 2016.

[39] Franz Hamilton, Alun L. Lloyd, and Kevin B. Flores. Hybrid modeling and prediction of dynamical systems. *PLoS Computational Biology*, 13, 2017.

[40] Liang Zhang, Gang Wang, and Georgios B. Giannakis. Real-time power system state estimation and forecasting via deep unrolled neural networks. *IRE Transactions on Audio*, 67(15):4069–4077, August 2019.

[41] Kun Yao, John E. Herr, David W. Toth, Ryker Mckintyre, and John Parkhill. The tensormol-0.1 model chemistry: a neural network augmented with long-range physics. *Chem. Sci.*, 9:2261–2269, 2018.

[42] Roberto Paolucci, Filippo Gatti, Maria Infantino, Chiara Smerzini, Ali Ozcebe, and Marco Stupazzini. Broadband ground motions from 3d physics-based numerical simulations using artificial neural networks. *Bulletin of the Seismological Society of America*, 108, 02 2018.

[43] Alaeddin Malek and R. Shekari Beidokhti. Numerical solution for high order differential equations using a hybrid neural network - optimization method. *Appl. Math. Comput.*, 183:260–271, 2006.

[44] Sindre Stenen Blakseth. Introducing CoSTA: A deep neural network enabled approach to improving physics-based numerical simulations. Master's thesis, NTNU, 2021.

[45] Sindre Stenen Blakseth, Adil Rasheed, Trond Kvamsdal, and Omer San. Deep neural network enabled corrective source term approach to hybrid analysis and modeling. *Neural Networks*, 146:181–199, 2022.

[46] Alfio Quarteroni. *Numerical models for differential problems*. Springer International Publishing, 3rd edition, 2017.

[47] Susanne C. Brenner and L. Ridgway Scott. *The mathematical theory of finite element methods*. Springer, New York, 3rd edition, 2008.

[48] Hasmah Mansor, Muhammad Helmy Abdul Shukor, Siti Sarah Meskam, Nur Quraisyia Aqilah Mohd Rusli, and Nasiha Sakinah Zamery. Body temperature measurement for remote health monitoring system. In *2013 IEEE International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA)*, pages 1–5, 2013.

[49] Amir Shahidi, Lokesh A. Gupta, Andrew Kovacs, and Dimitrios Peroulis. Wireless temperature and vibration sensor for real-time bearing condition monitoring. In *2013 IEEE MTT-S International Microwave Symposium Digest (MTT)*, pages 1–4, 2013.

[50] Hardikkumar Prajapati, Swapnil S. Salvi, Darshan Ravoori, and Ankur Jain. Measurement of the in-plane temperature field on the build plate during polymer extrusion additive manufacturing using infrared thermometry. *Polymer Testing*, 92:106866, 2020.

[51] D. Borthwick. *Introduction to Partial Differential Equations*. Universitext. Springer International Publishing, 2018.

[52] Richard Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49:1–23, 1943.

[53] William S. Slaughter. *The Linearized Theory of Elasticity*. Birkhäuser, Boston, MA, 2002.

[54] F. Irgens. *Continuum Mechanics*. Springer Berlin Heidelberg, 2008.

[55] Gaetano Fichera. *Existence Theorems in Elasticity*, pages 347–389. Springer Berlin Heidelberg, Berlin, Heidelberg, 1973.

[56] Norman G. Meyers and James Serrin. H = W. *Proceedings of the National Academy of Sciences of the United States of America*, 51 6:1055–6, 1964.

[57] George V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[58] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert L. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[59] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[60] Ismoilov Nusrat and Sung-Bong Jang. A comparison of regularization techniques in deep neural networks. *Symmetry*, 10(11), 2018.

[61] Patrick J. Roache. Code Verification by the Method of Manufactured Solutions . *Journal of Fluids Engineering*, 124(1):4–10, 2001.

[62] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[63] Andrew L. Maas, Awni Y. Hannum, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. *Journal of Machine Learning Research: W&CP*, 28, 2013.

[64] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[65] Peter Wriggers. *Nonlinear Finite Element Methods*. Springer Berlin, Heidelberg, 2008.

Appendix A
Final Solution Results for Two-Dimensional Experiments

In this chapter we present plots of final solutions and predictions of the two-dimensional tests. Only the mean prediction is plotted of DDM and CoSTA.

A.1 Dimensional Reduction on Heat Equation

Plots of the final solutions of the second setup with dimensional reduction on the heat equation are shown in Figures A.1.1 to A.1.8.

A.2 Linear Elasticity Equation

Plots of the final solutions of the elasticity equation with exact source term are shown in Figures A.2.1 to A.2.6, and with zero source term are shown in Figures A.2.7 to A.2.12.

A.3 Dimensional Reduction on Linear Elasticity Equation

Plots of the final solutions with dimensional reduction on the elasticity equation are shown in Figures A.3.1 to A.3.6.

A.4 Nonlinear Elasticity Equation

Plots of the final solutions of the nonlinear elasticity equation are shown in Figures A.4.1 to A.4.6.

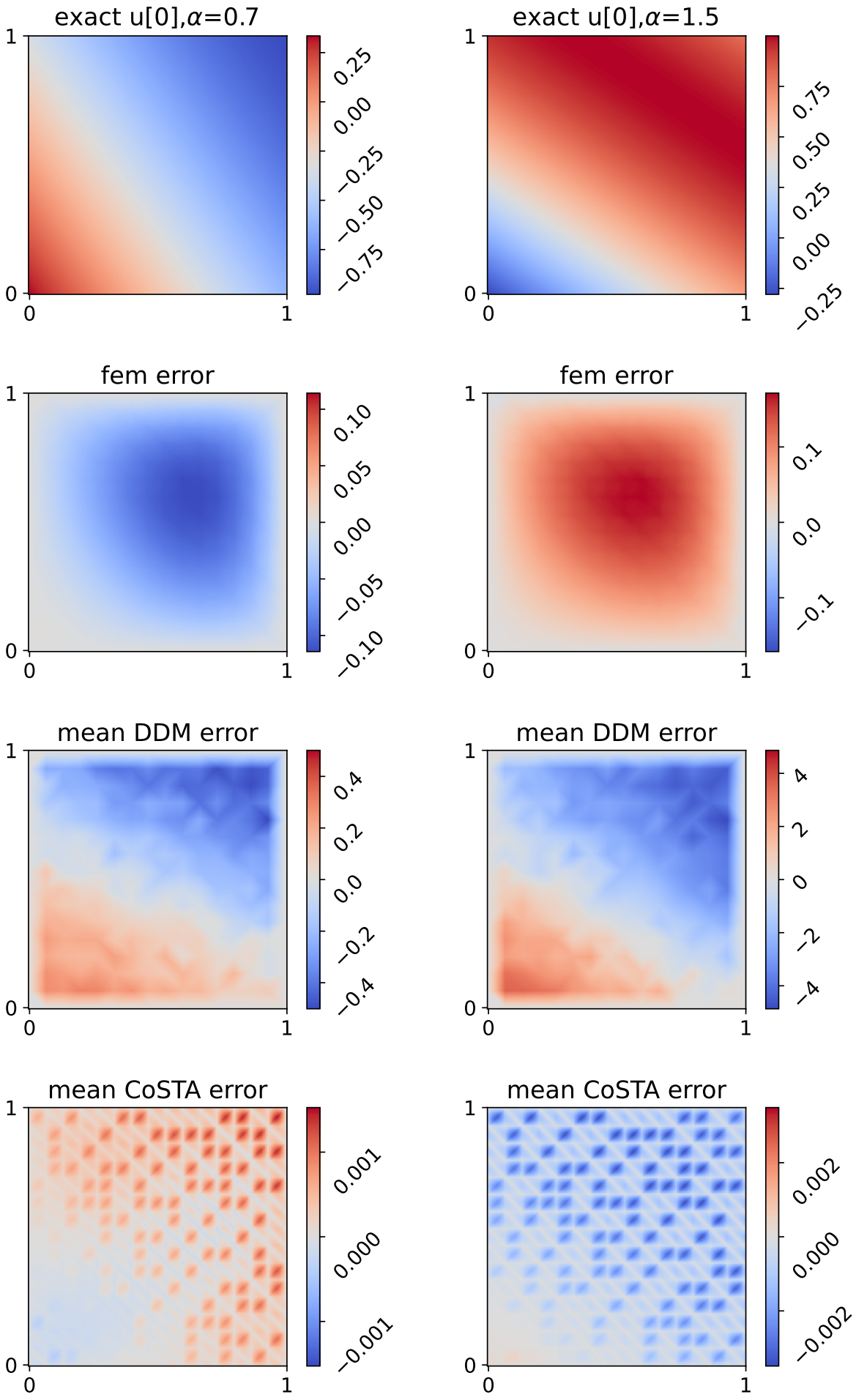


Figure A.1.1: Solution *d1*, interpolation

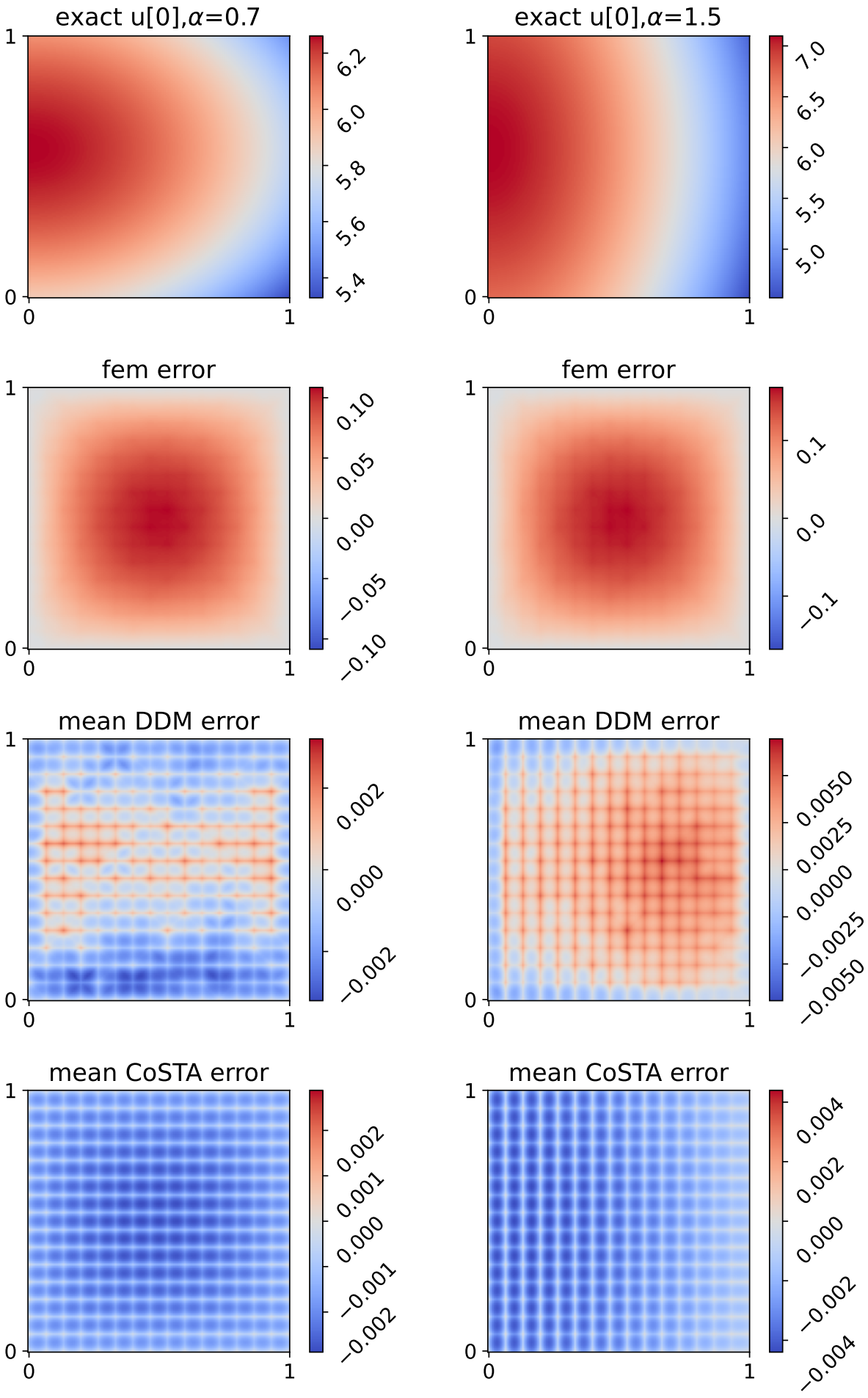


Figure A.1.2: Solution *d2*, interpolation

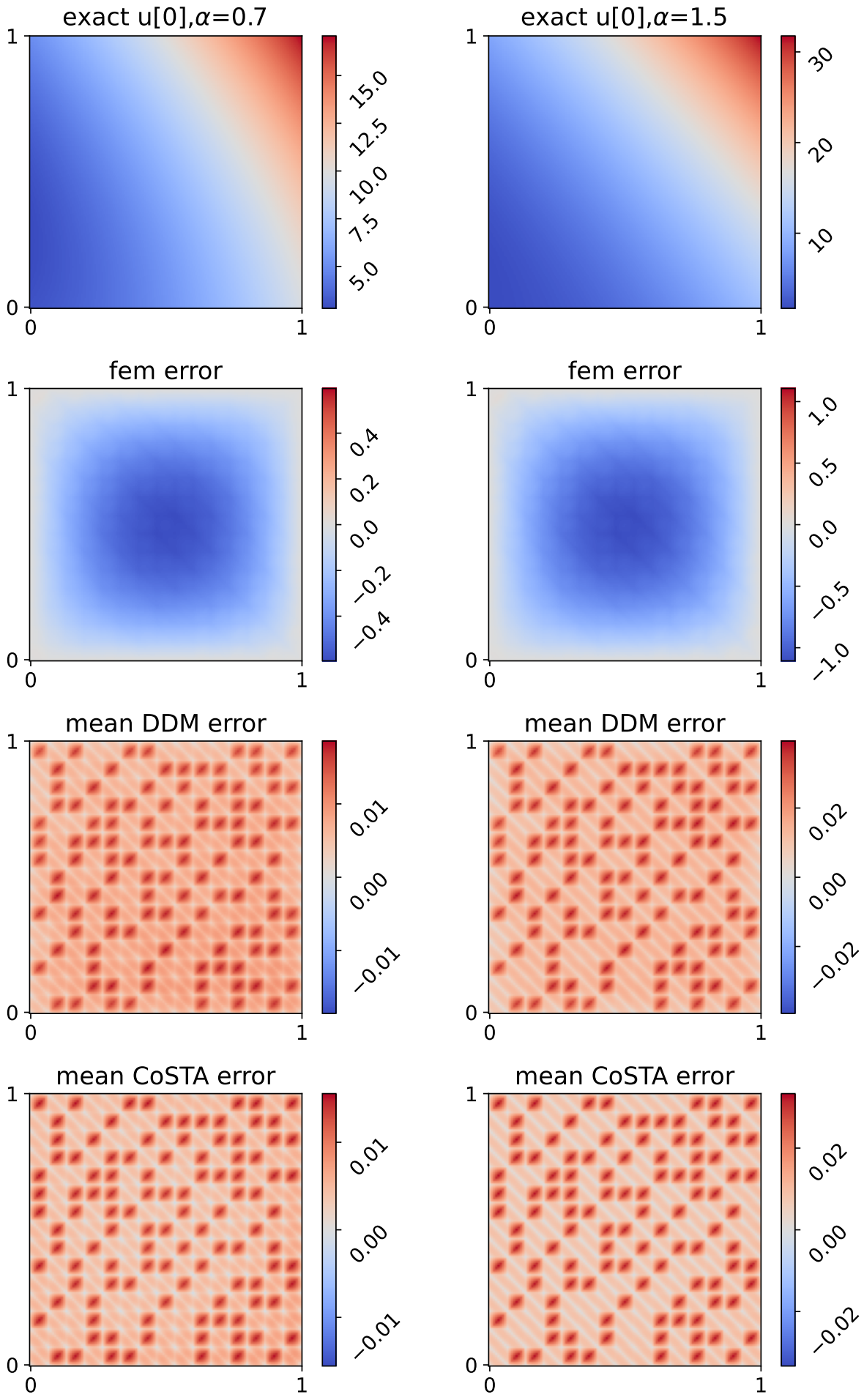


Figure A.1.3: Solution *d3*, interpolation

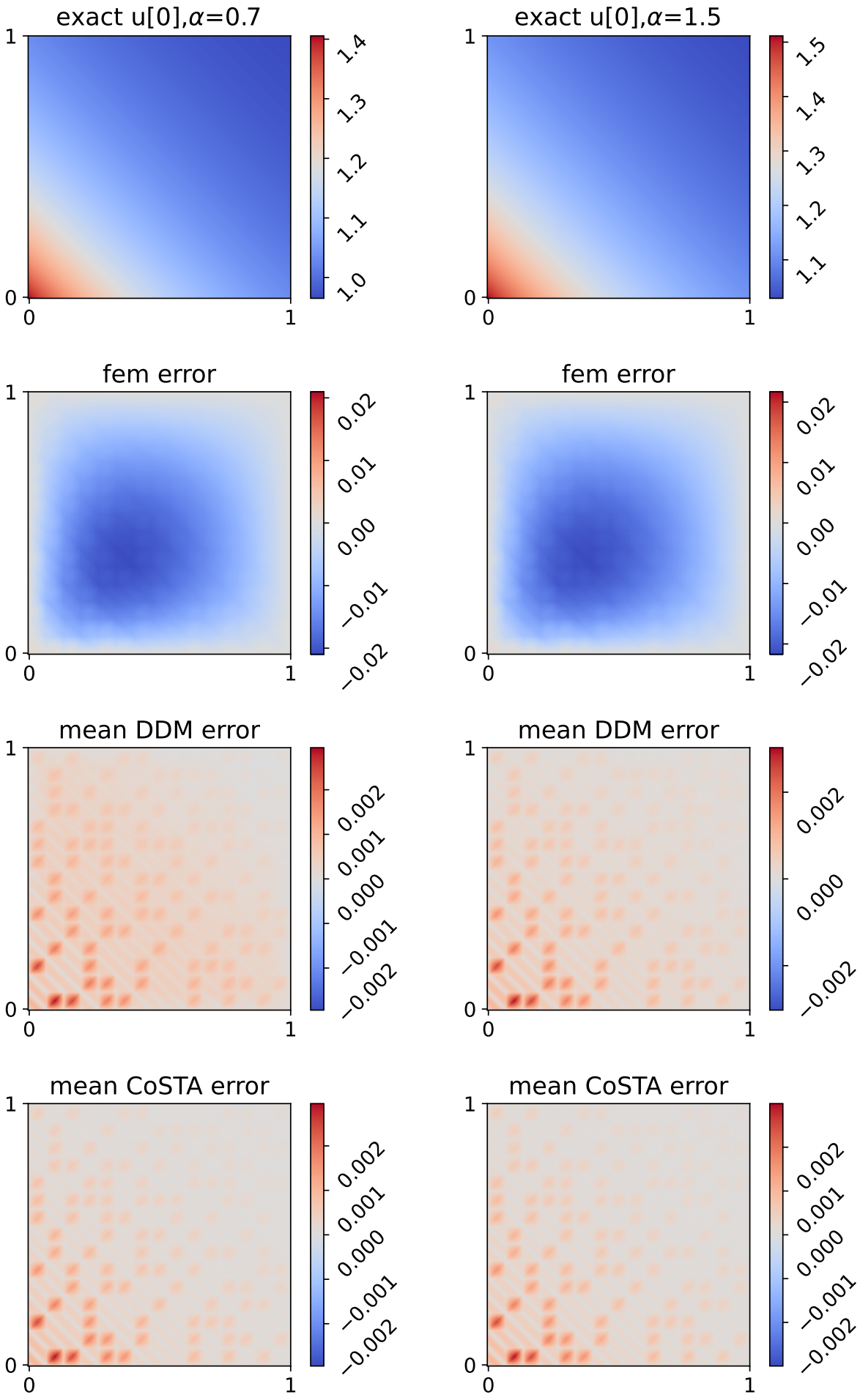


Figure A.1.4: Solution *d4*, interpolation

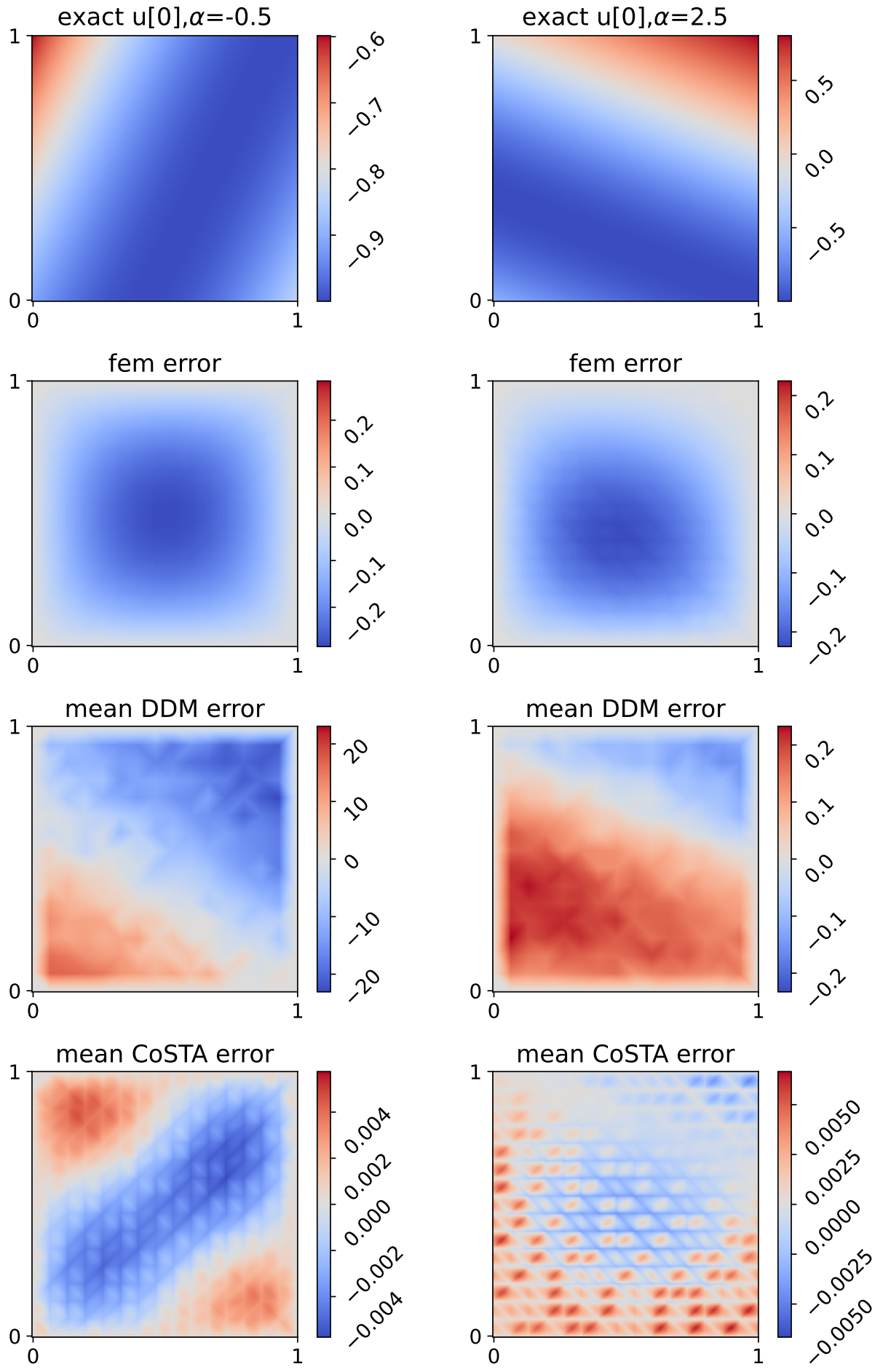


Figure A.1.5: Solution *d1*, extrapolation

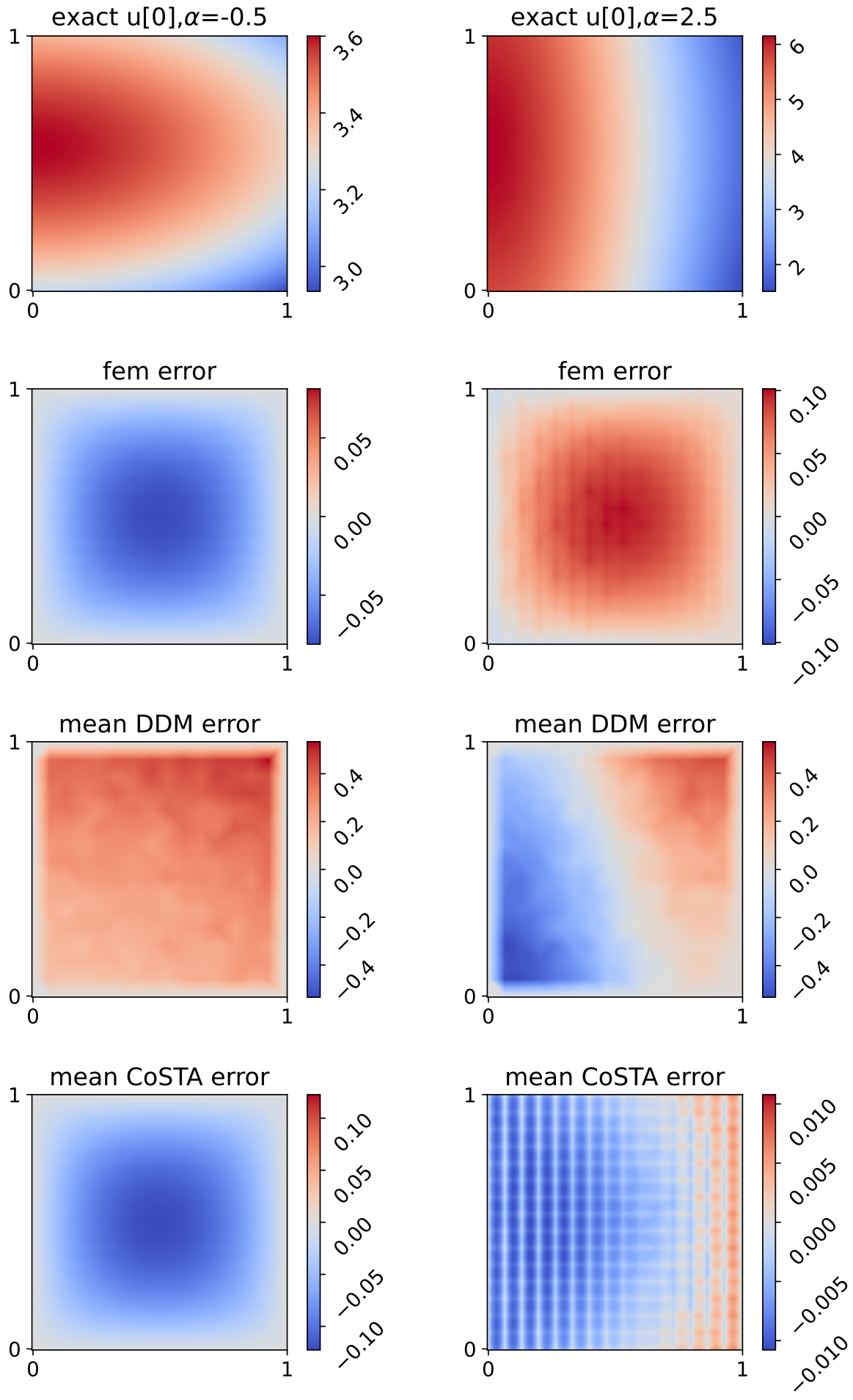


Figure A.1.6: Solution *d2*, extrapolation

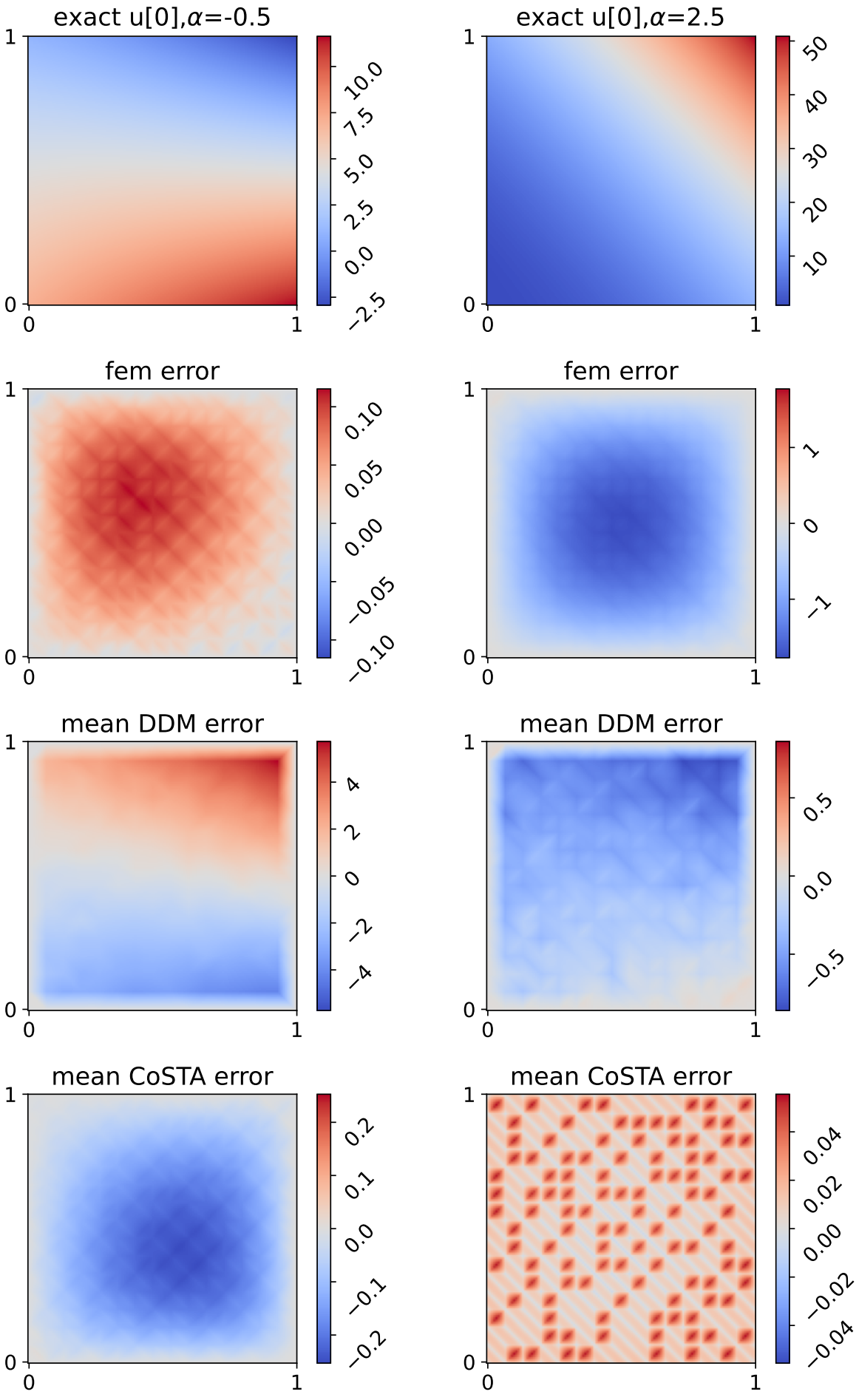


Figure A.1.7: Solution *d3*, extrapolation

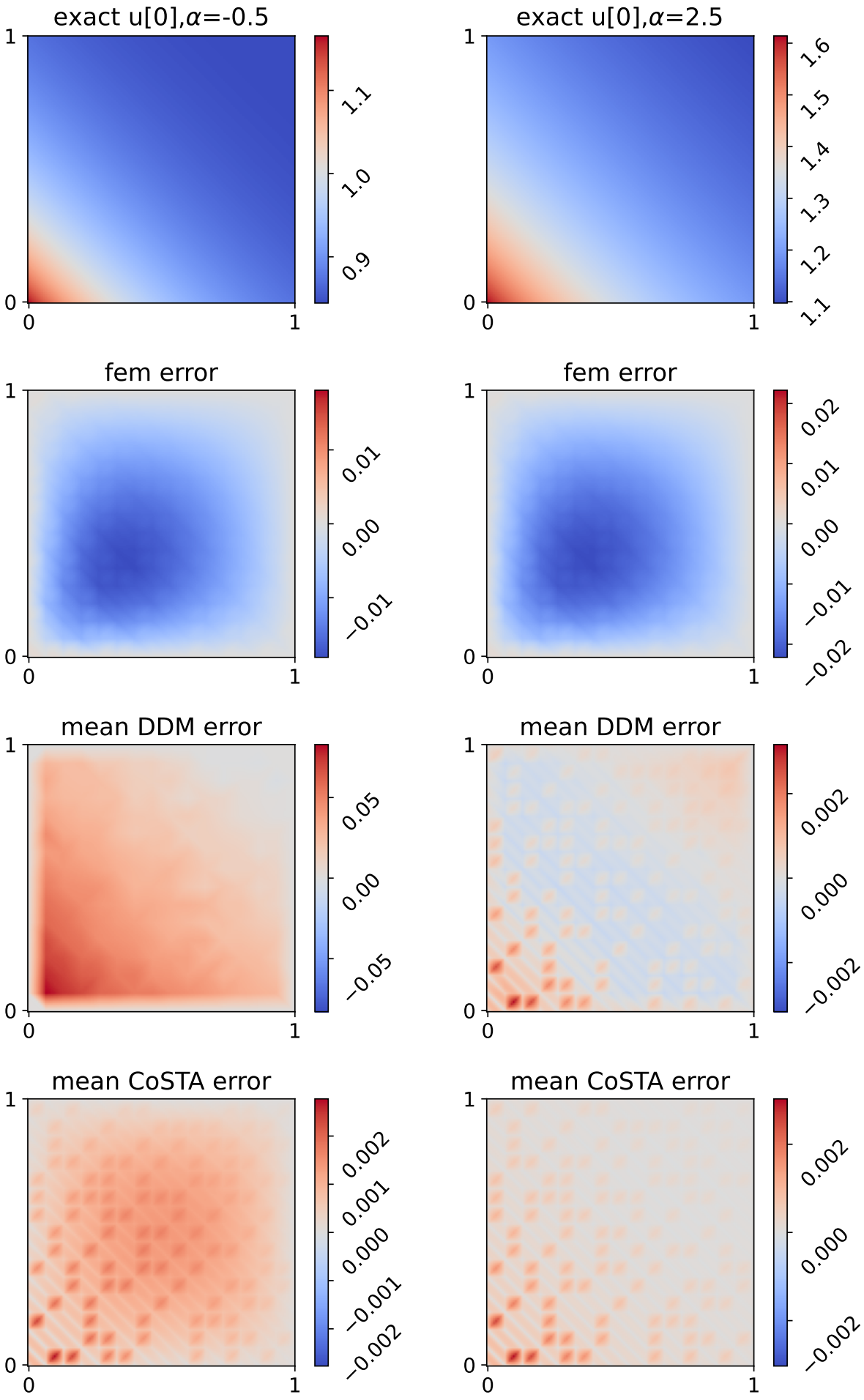


Figure A.1.8: Solution *d4*, extrapolation

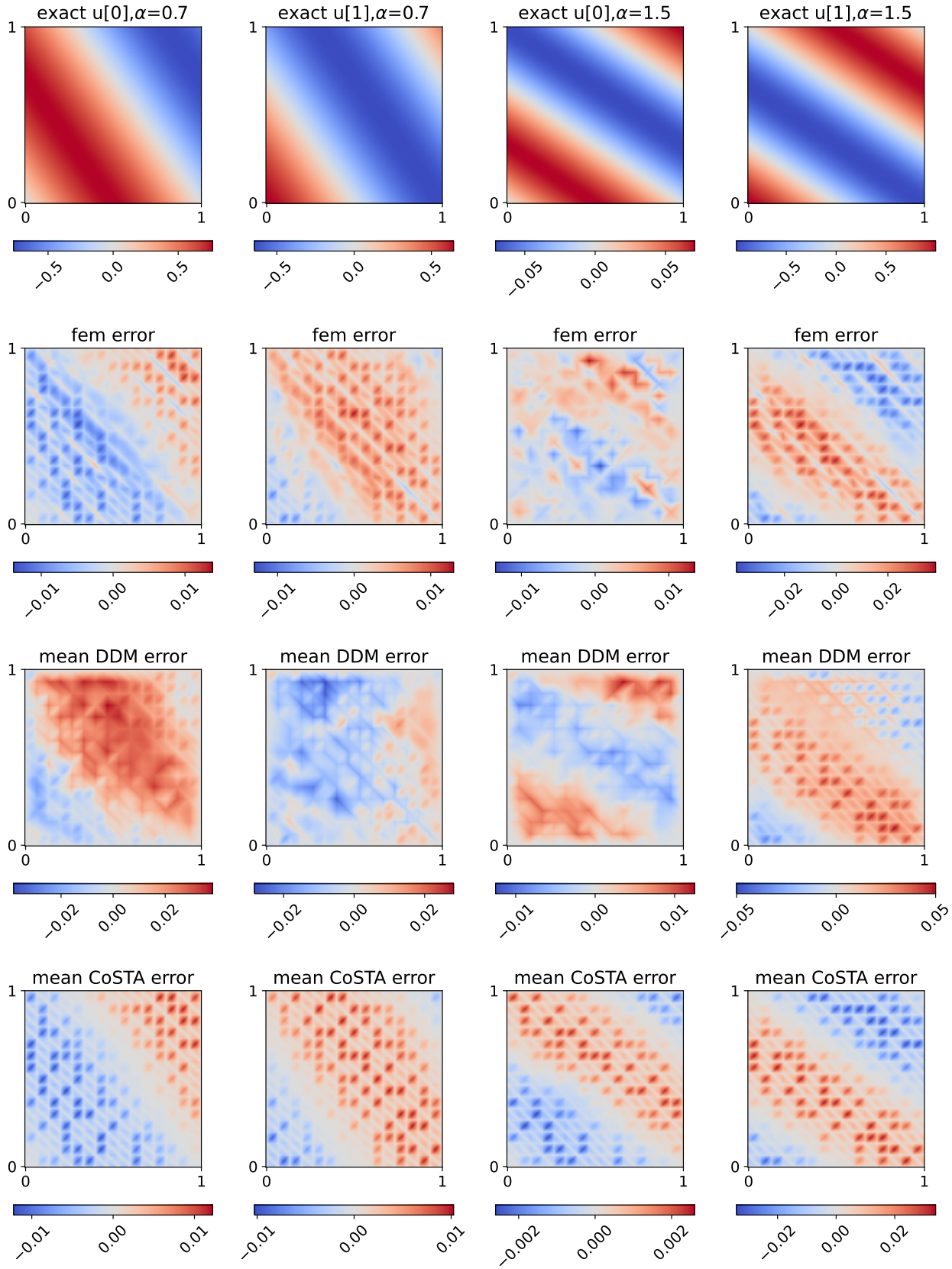


Figure A.2.1: Solution *e1* with correct source term, interpolation

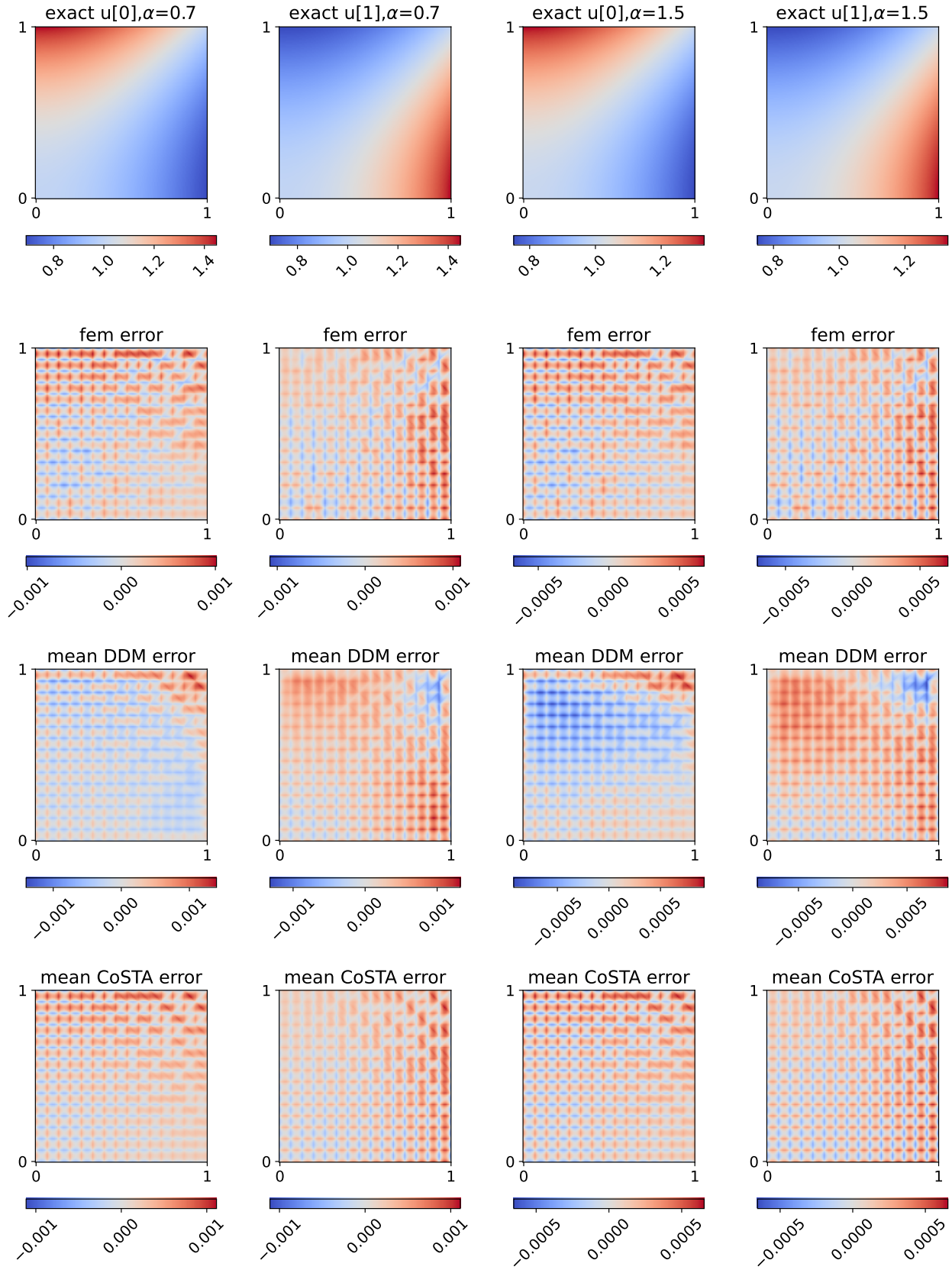


Figure A.2.2: Solution *e2* with correct source term, interpolation

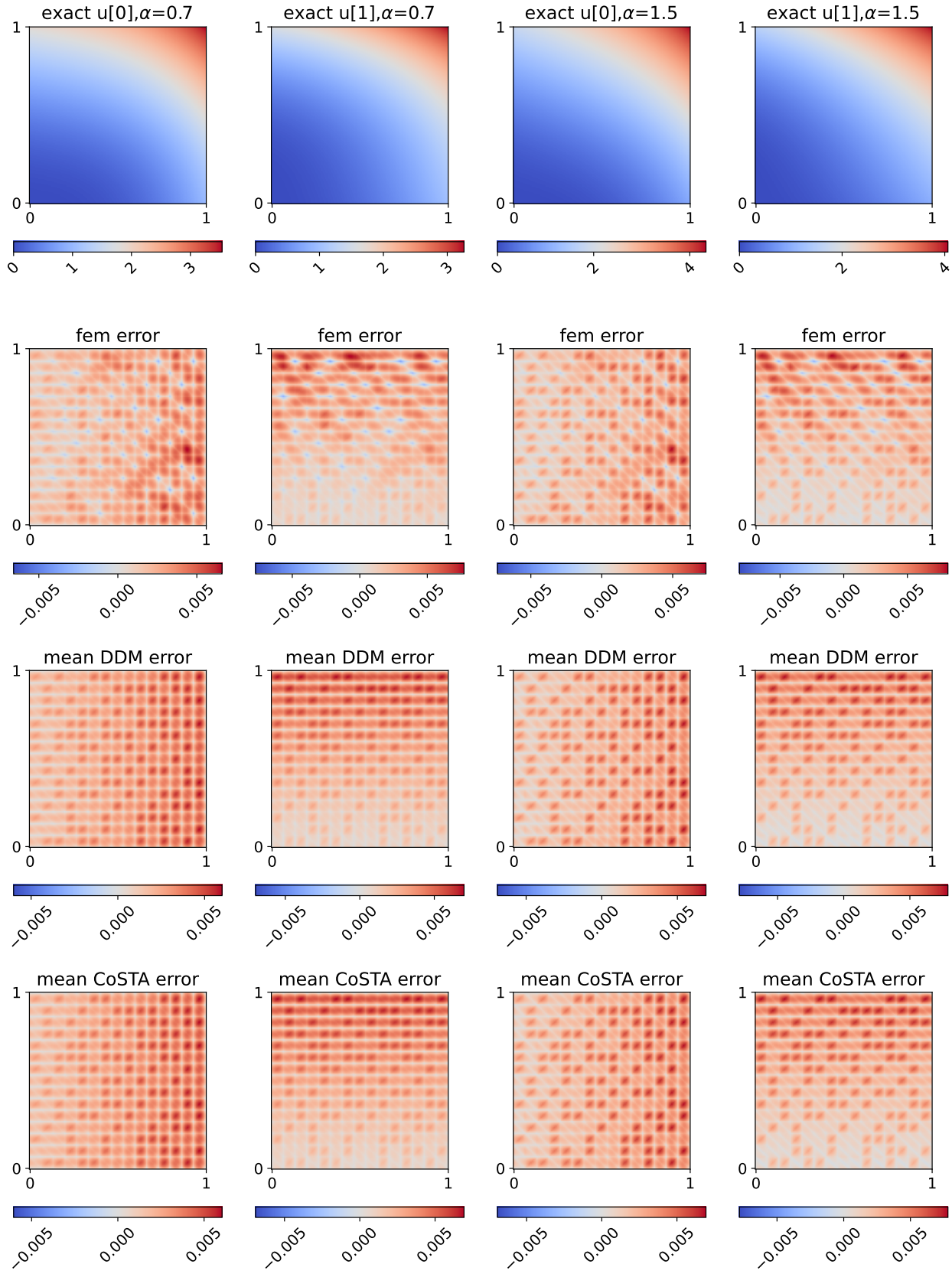


Figure A.2.3: Solution *e3* with correct source term, interpolation

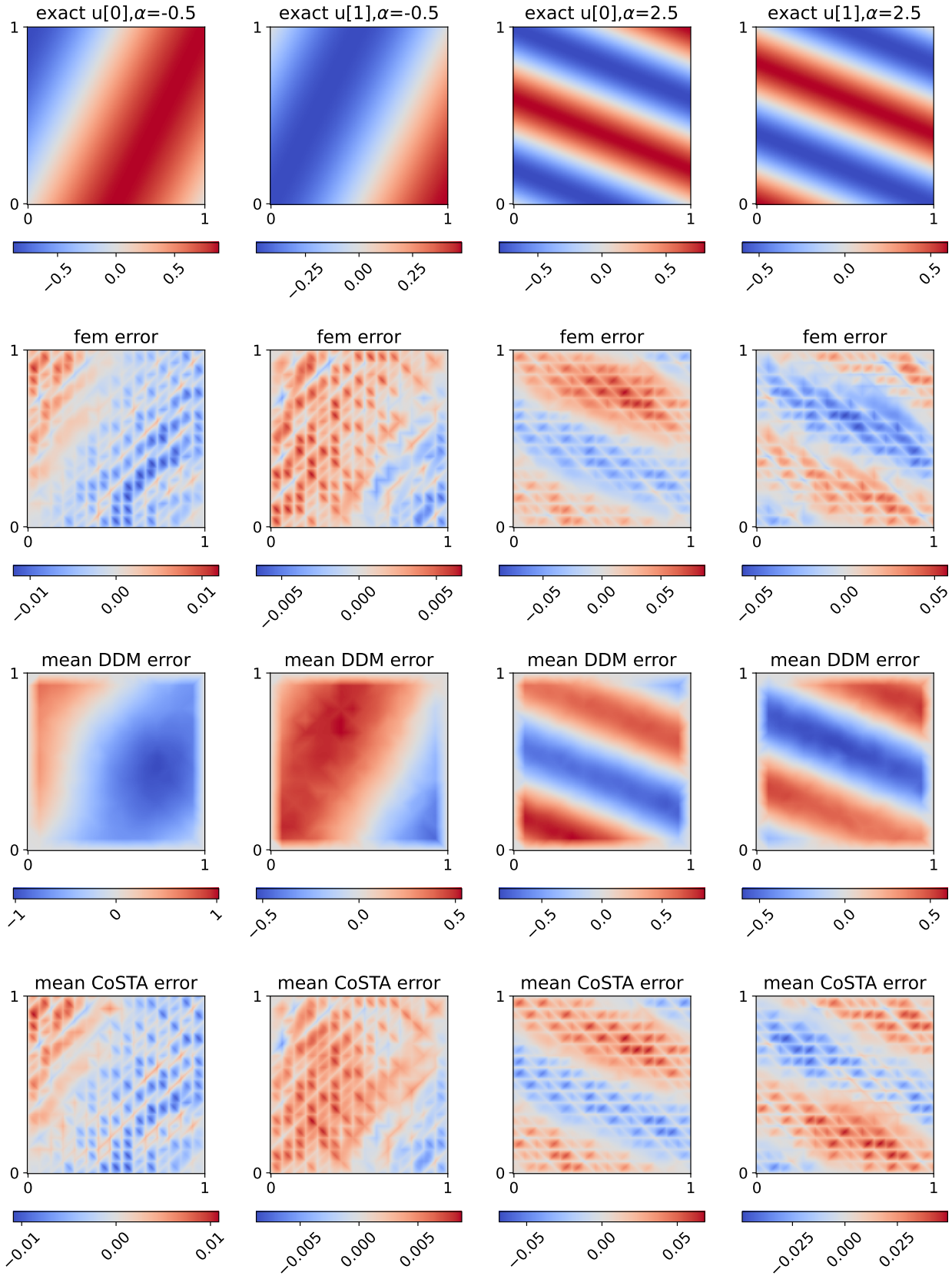


Figure A.2.4: Solution *e1* with correct source term, extrapolation

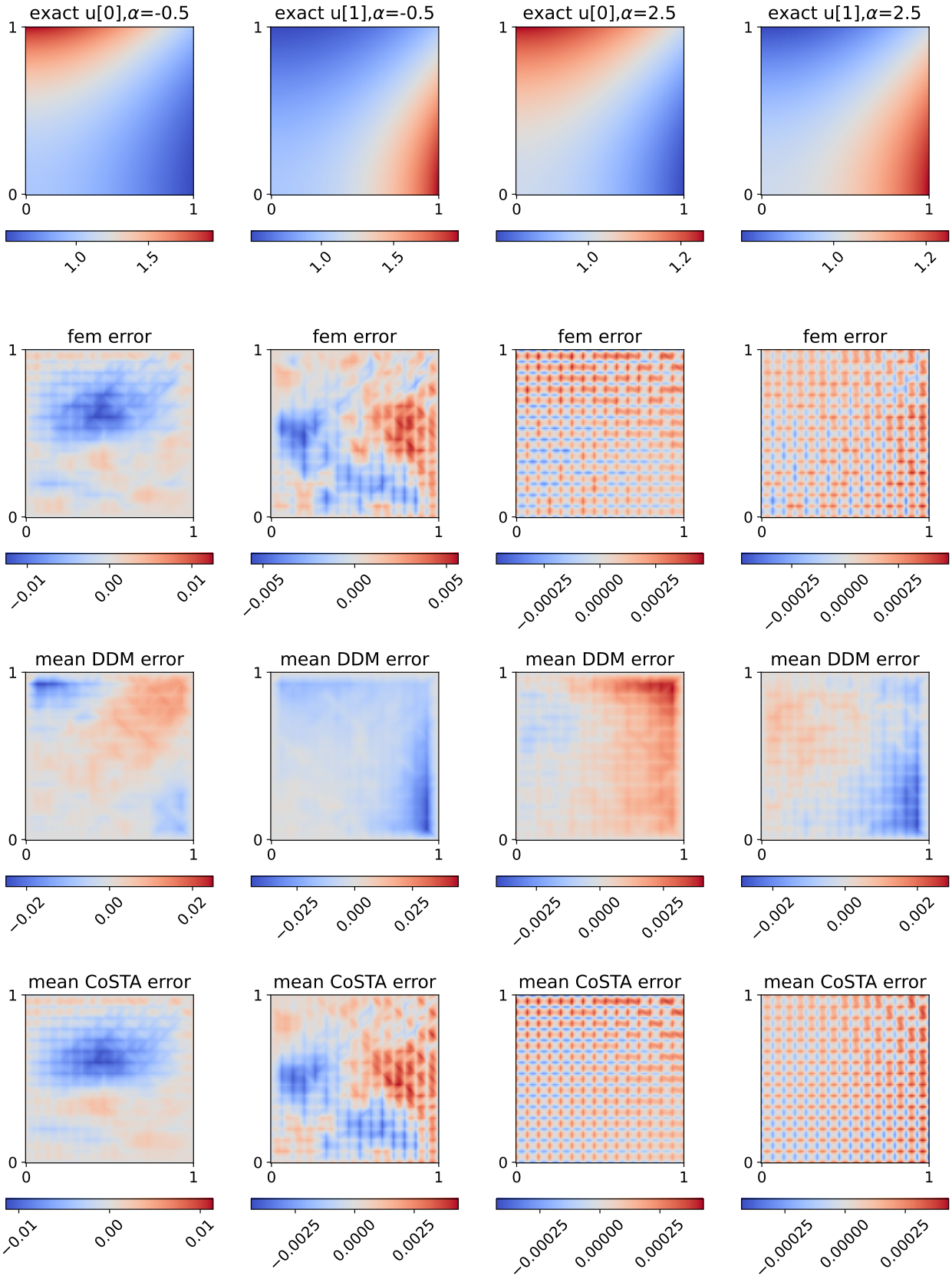


Figure A.2.5: Solution *e2* with correct source term, extrapolation

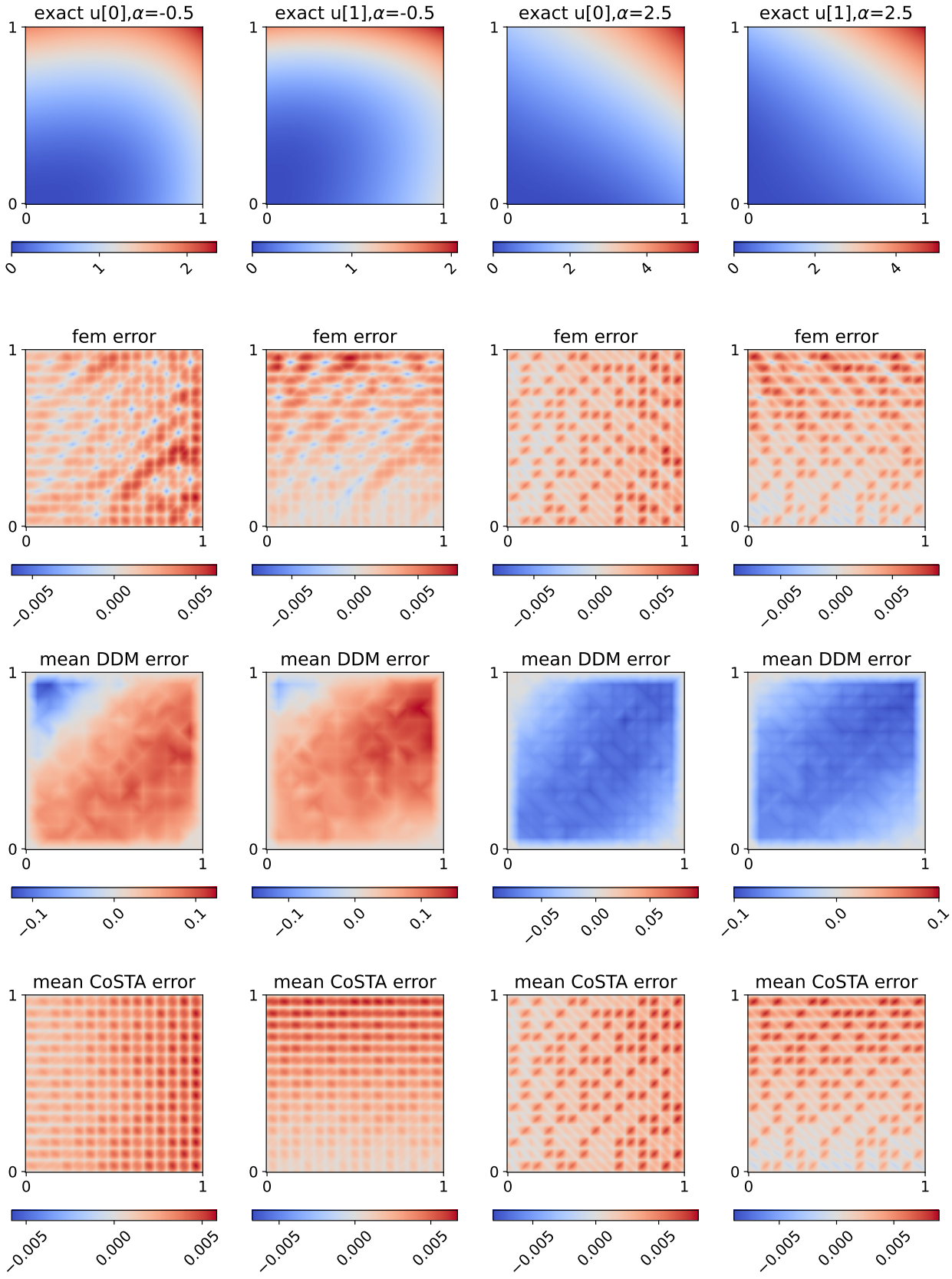


Figure A.2.6: Solution *e3* with correct source term, extrapolation

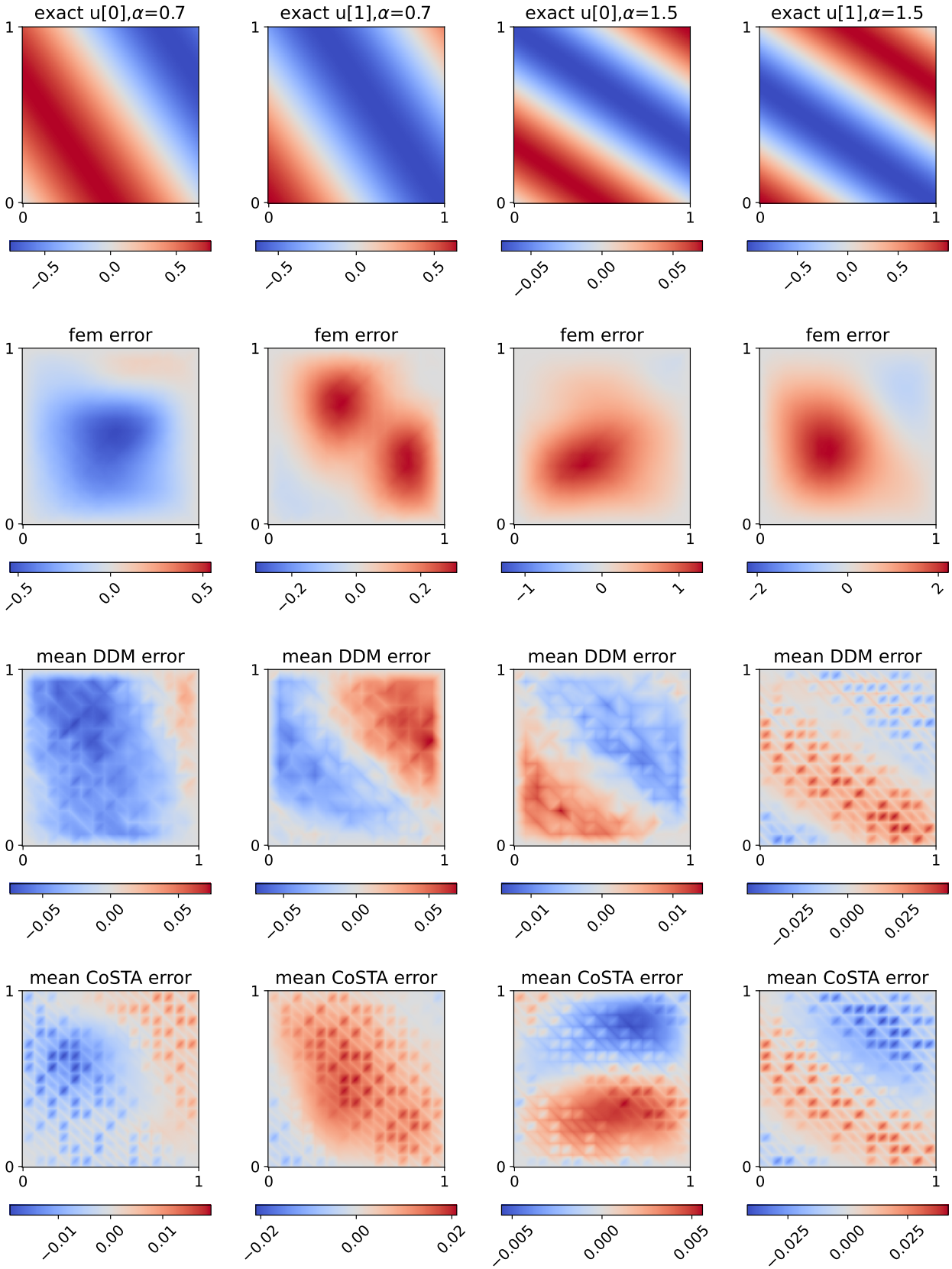


Figure A.2.7: Solution *e1* with zero source term, interpolation

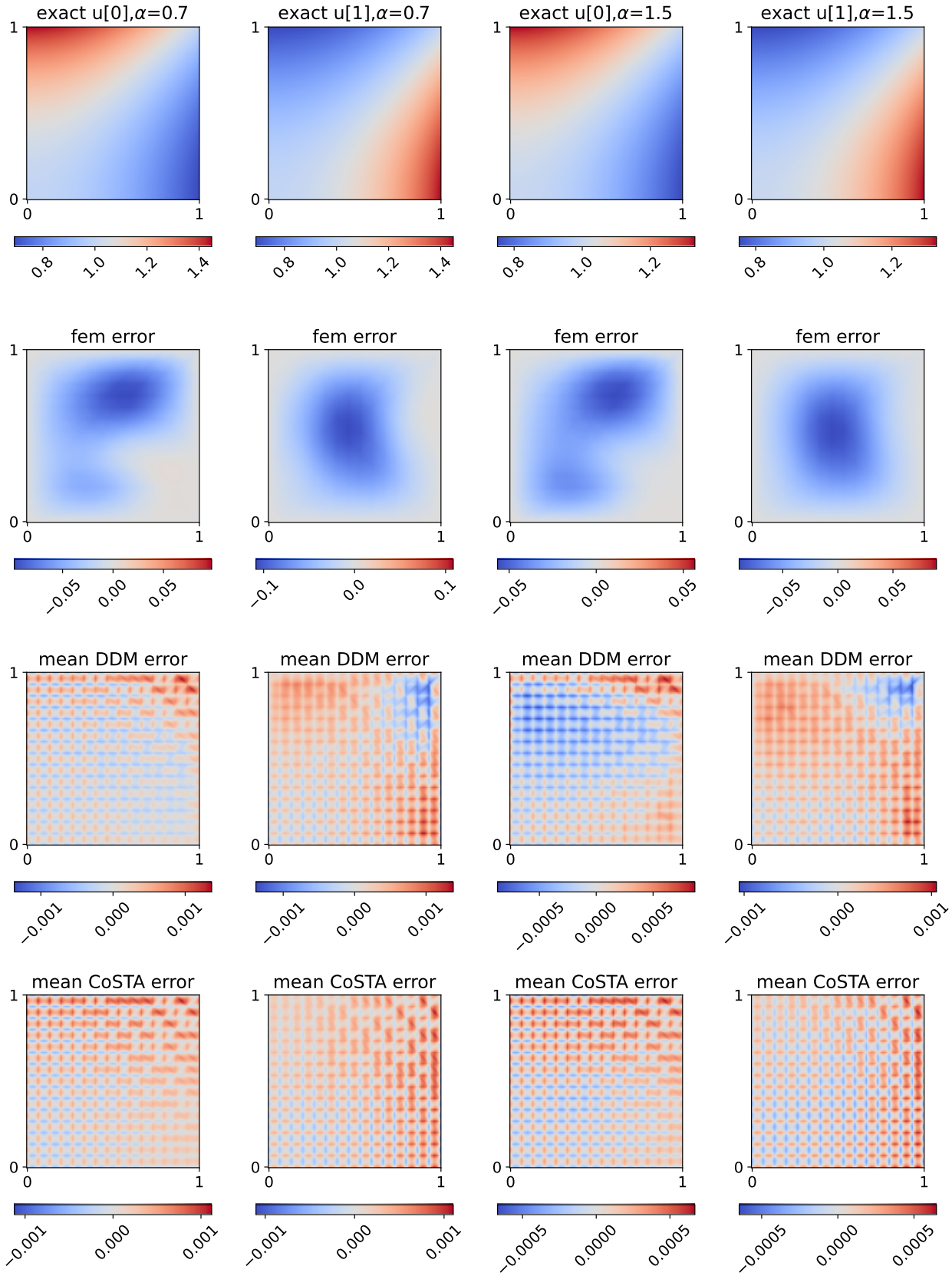


Figure A.2.8: Solution *e2* with zero source term, interpolation

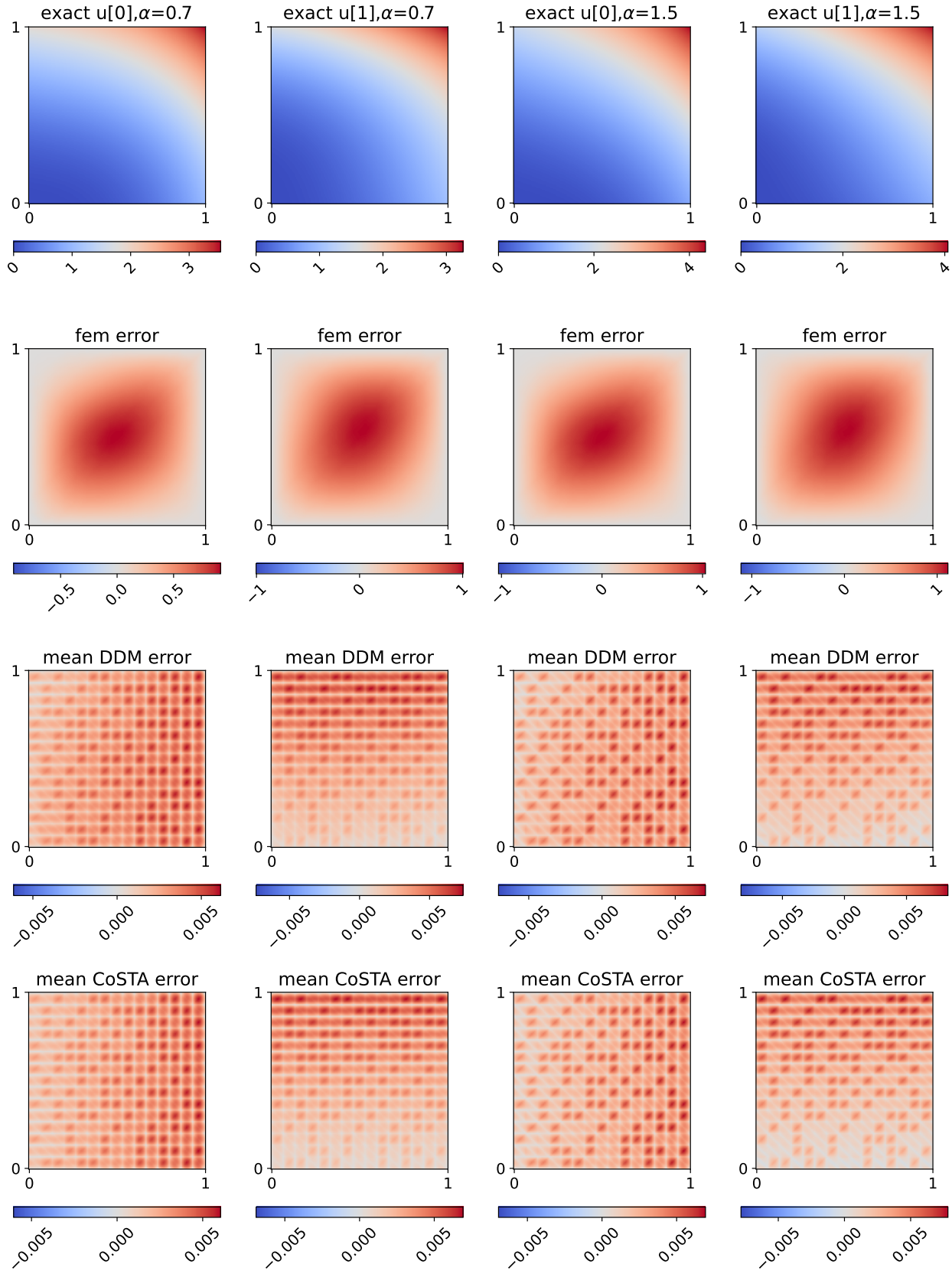


Figure A.2.9: Solution *e3* with zero source term, interpolation

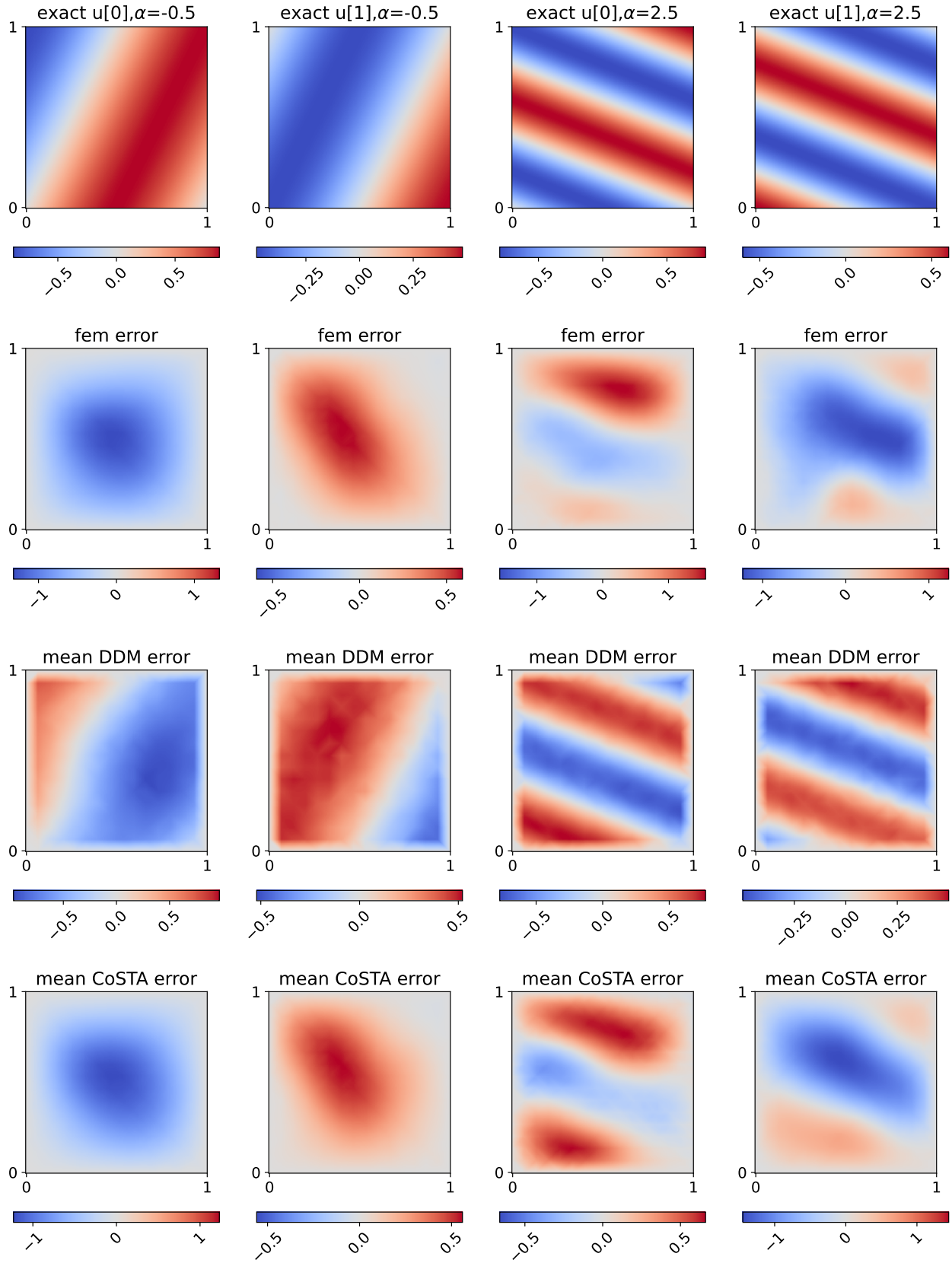


Figure A.2.10: Solution *e1* with zero source term, extrapolation

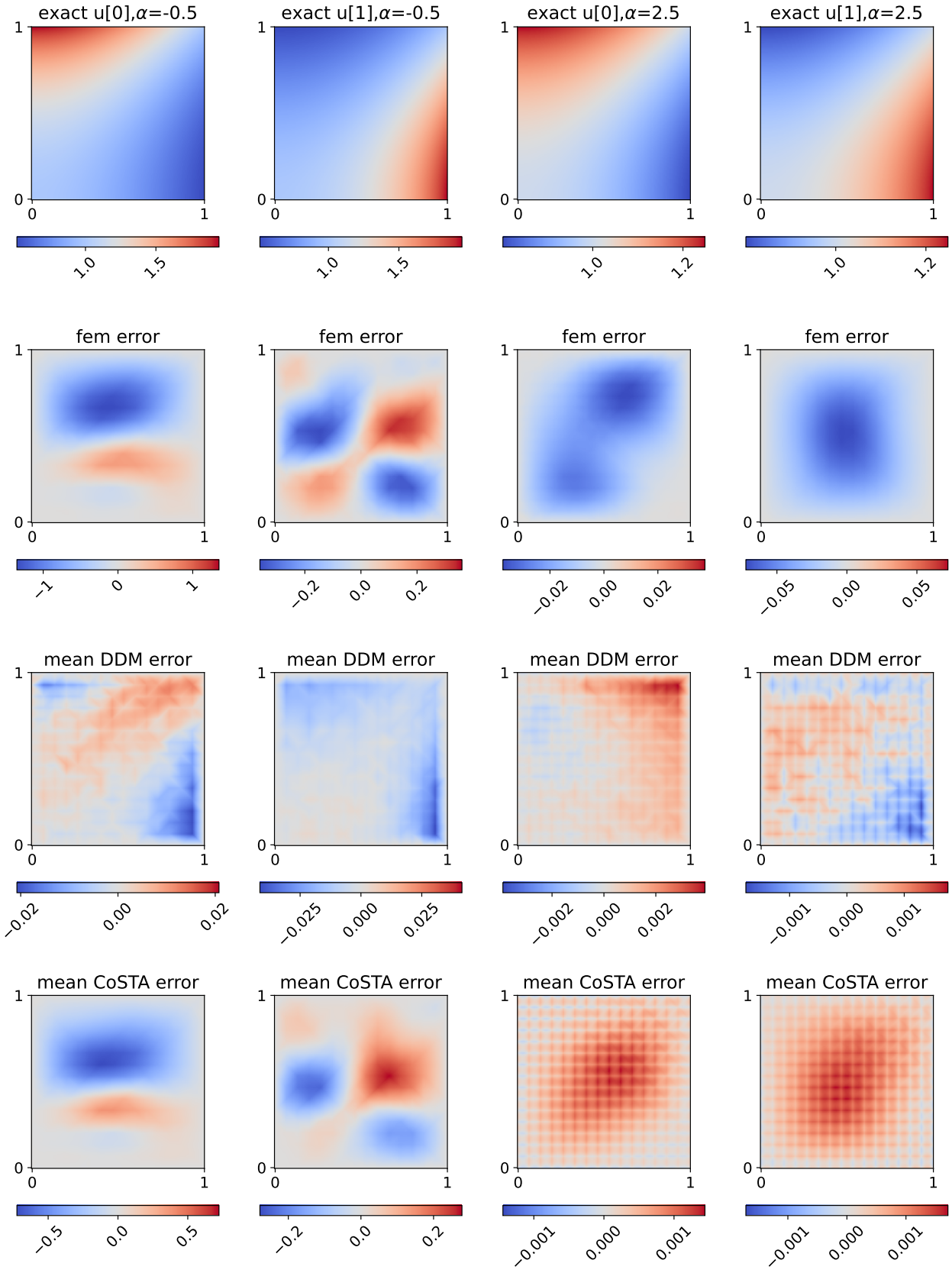


Figure A.2.11: Solution *e2* with zero source term, extrapolation

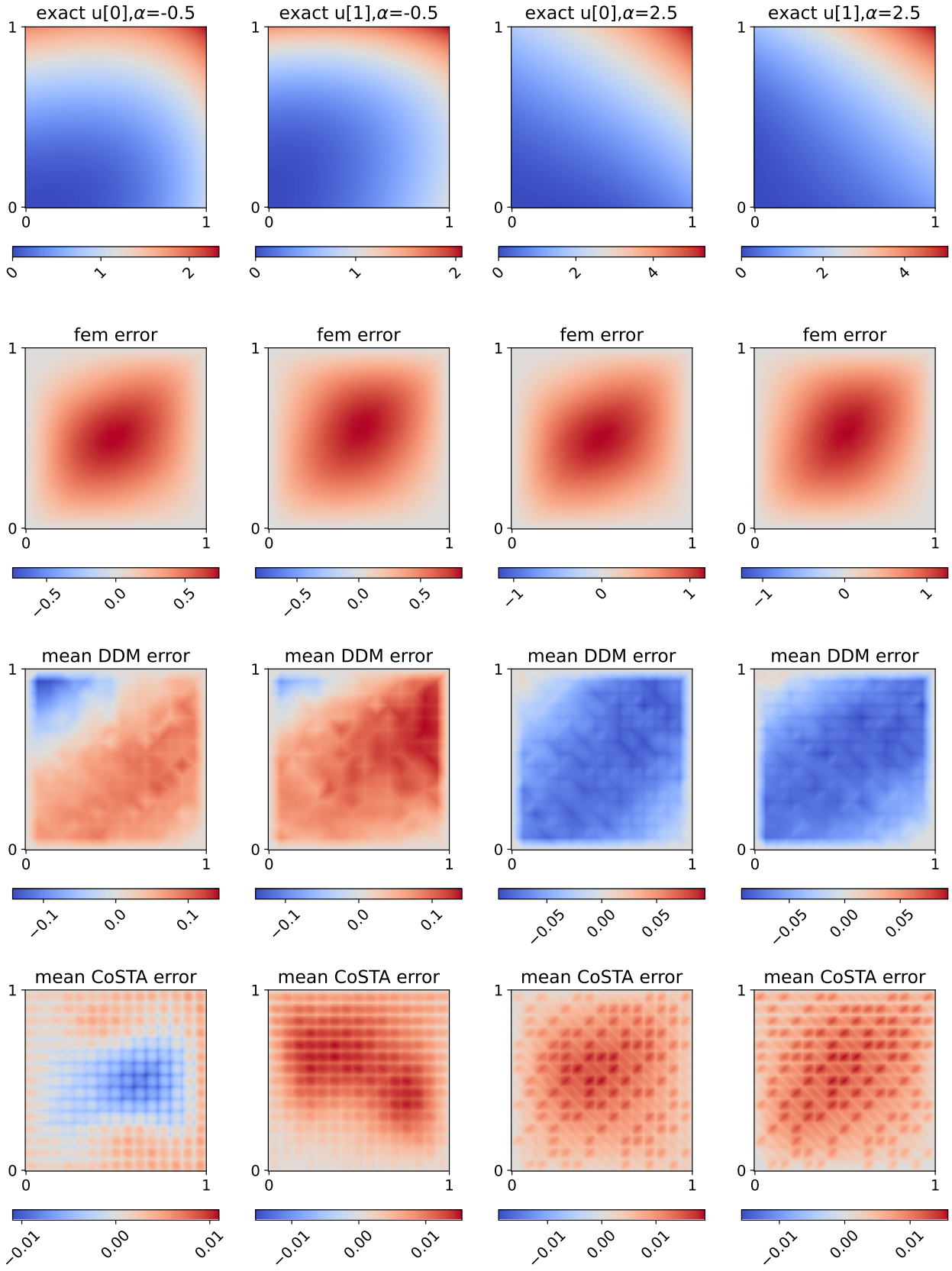


Figure A.2.12: Solution *e3* with zero source term, extrapolation

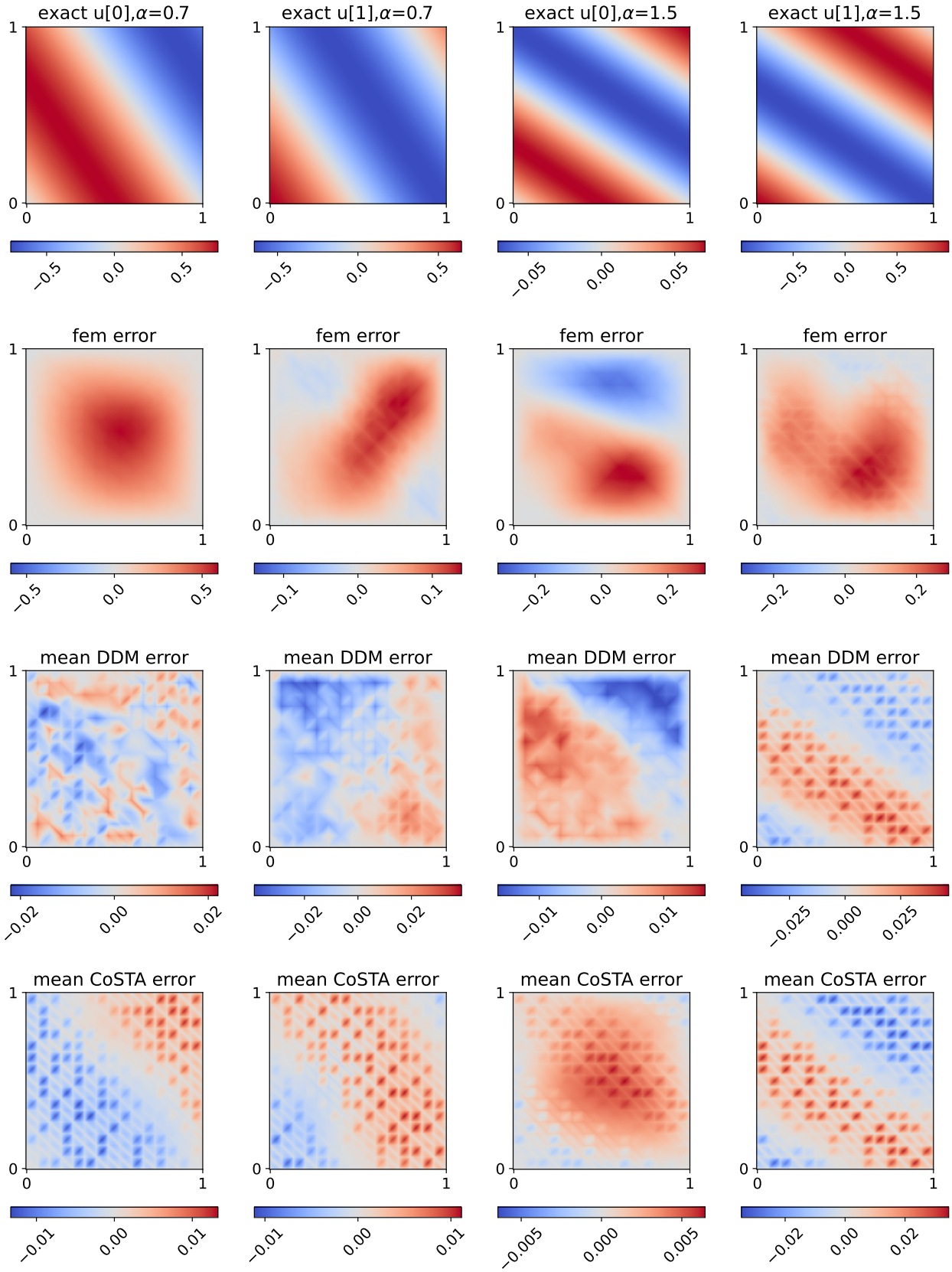


Figure A.3.1: Solution *ed1*, interpolation

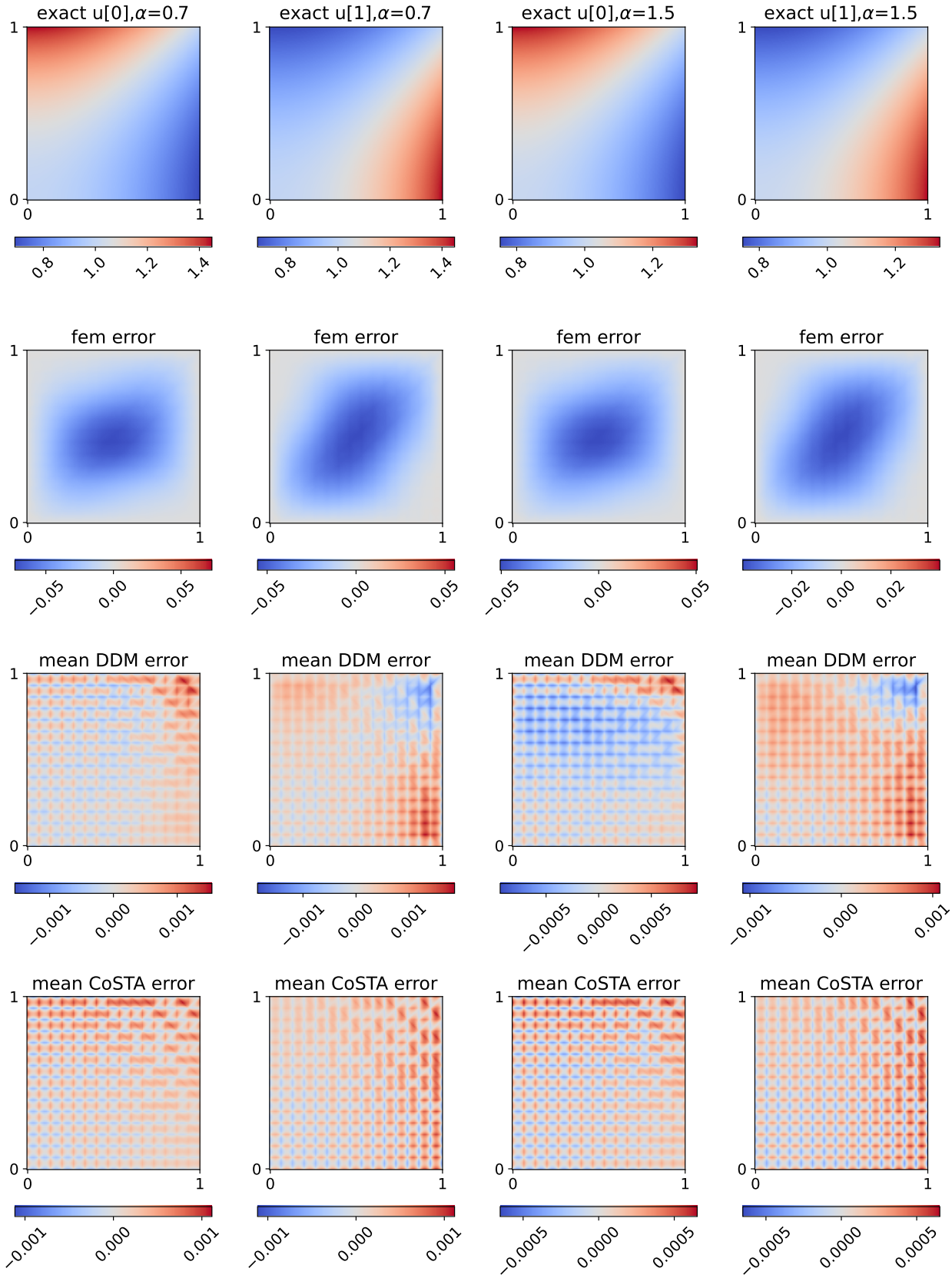


Figure A.3.2: Solution *ed2*, interpolation

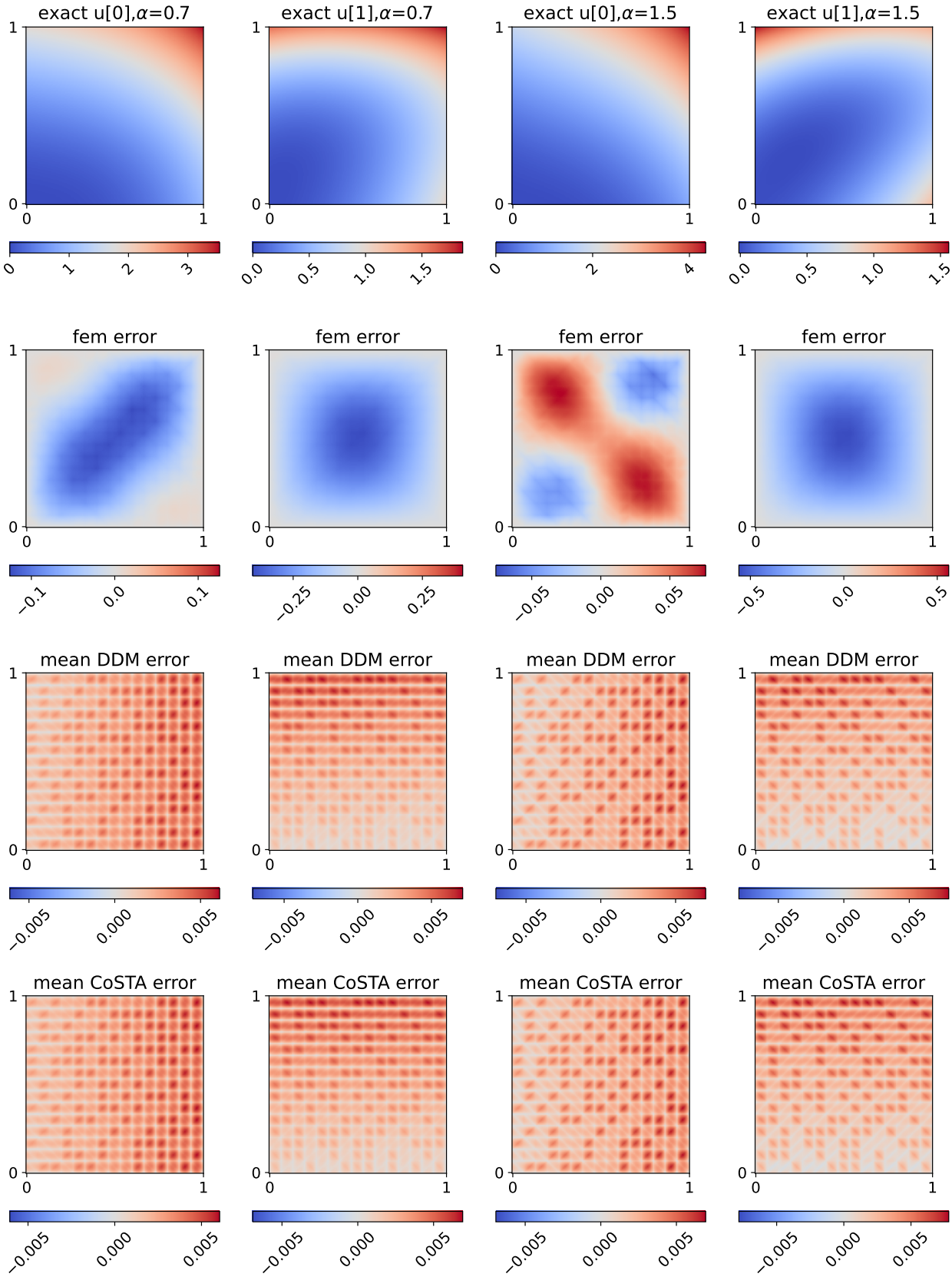


Figure A.3.3: Solution *ed3*, interpolation

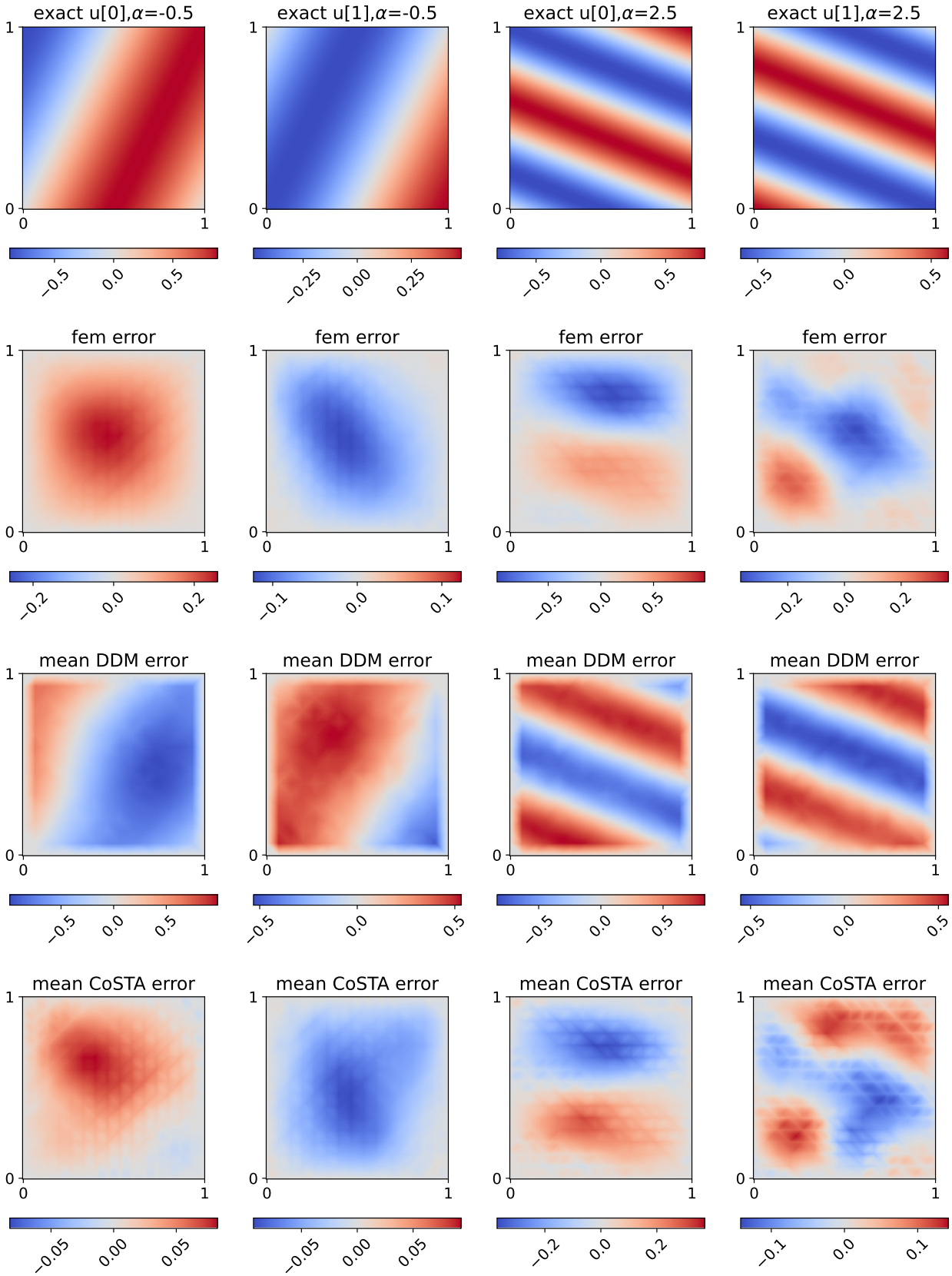


Figure A.3.4: Solution *ed1*, extrapolation

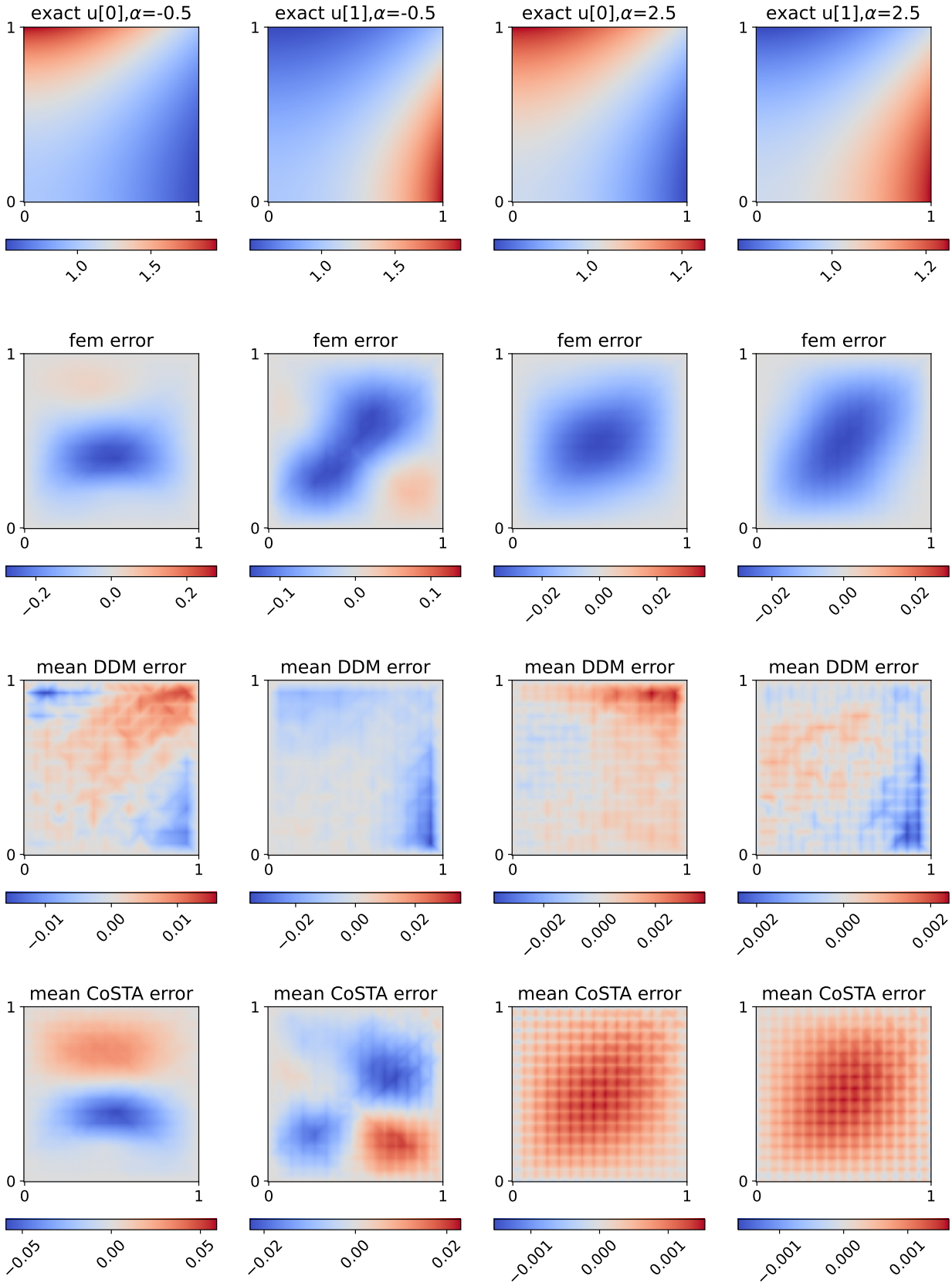


Figure A.3.5: Solution *ed2*, extrapolation

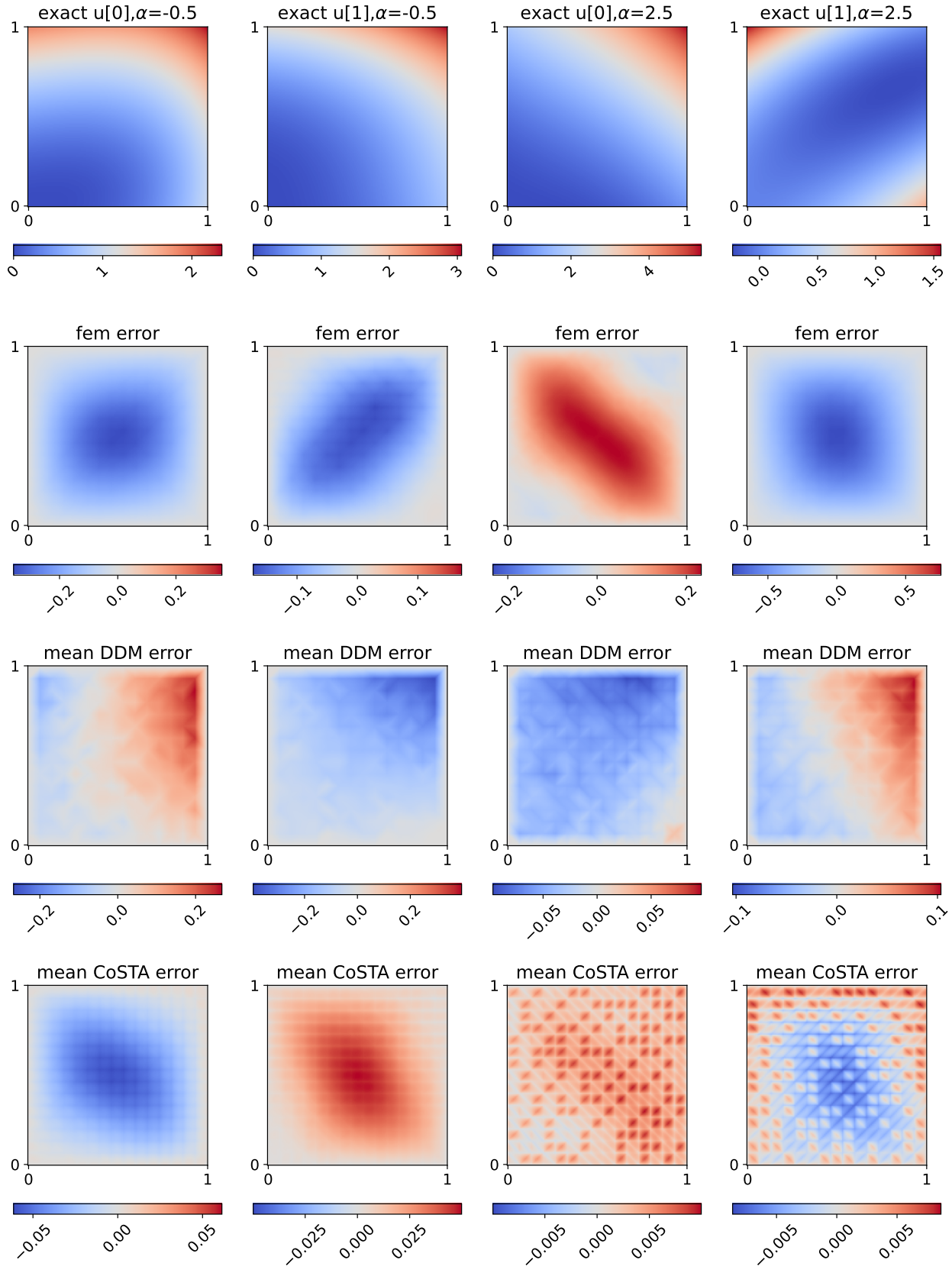


Figure A.3.6: Solution *ed3*, extrapolation

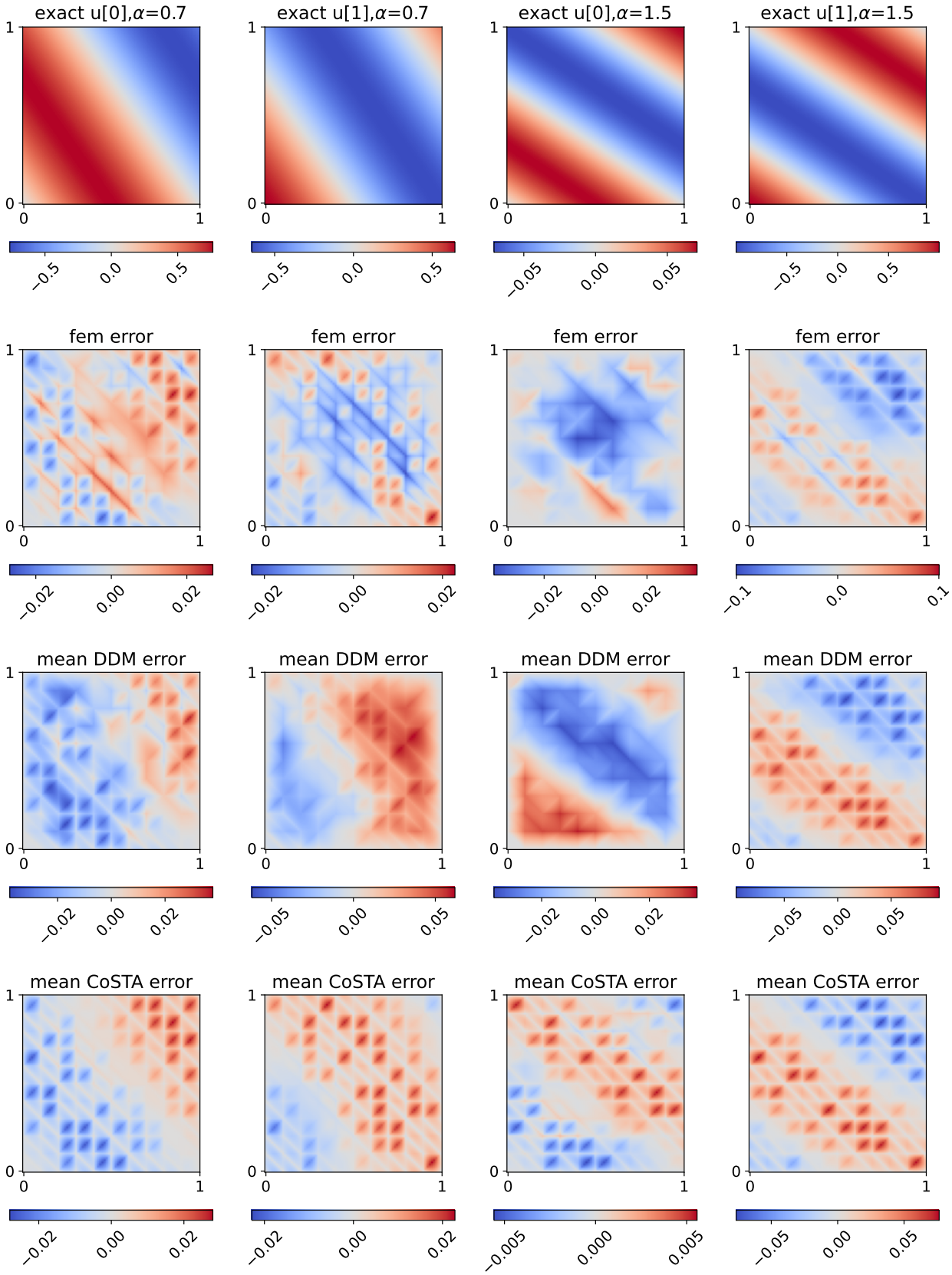


Figure A.4.1: Solution *n1*, interpolation

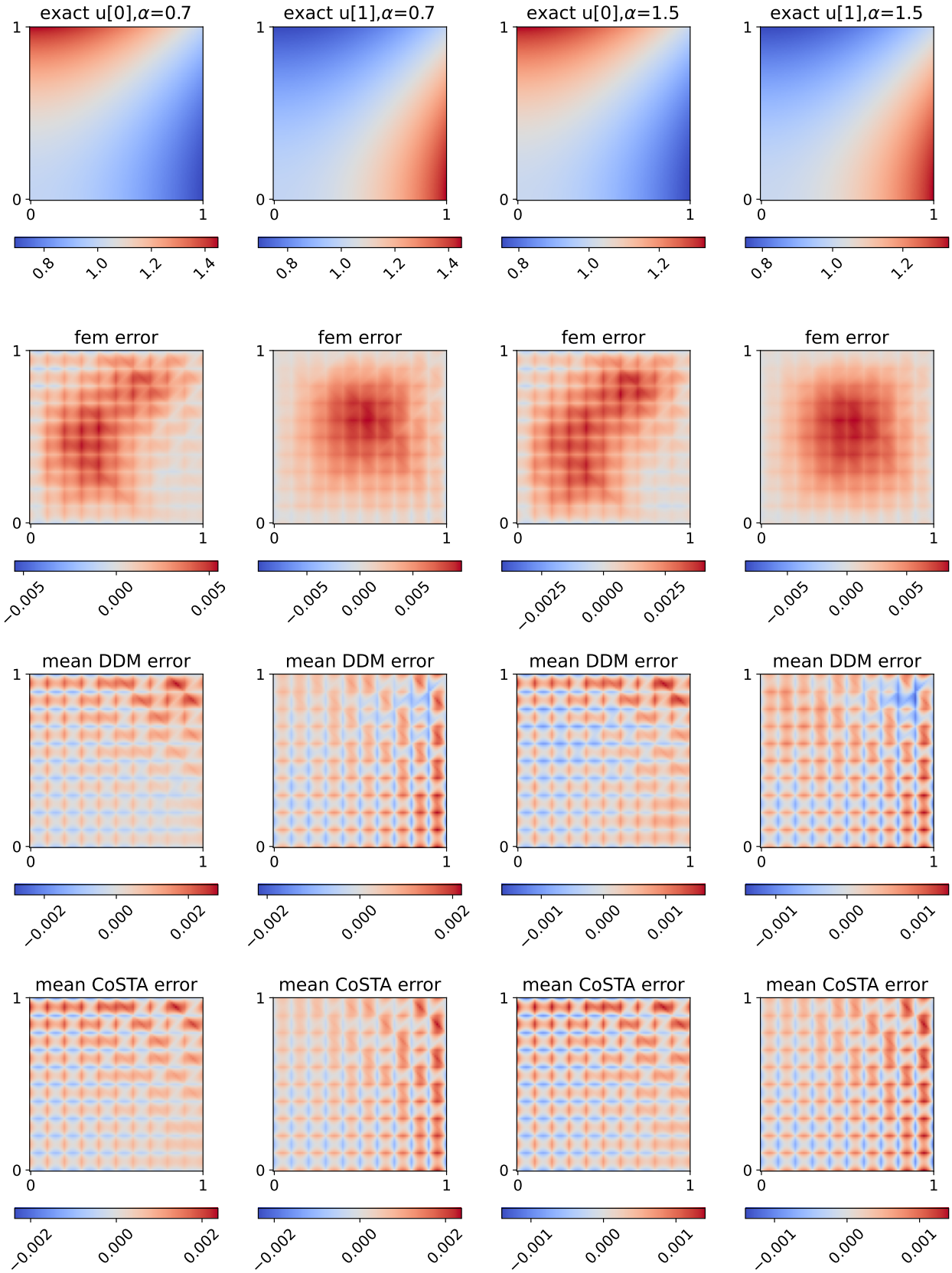


Figure A.4.2: Solution *n2*, interpolation

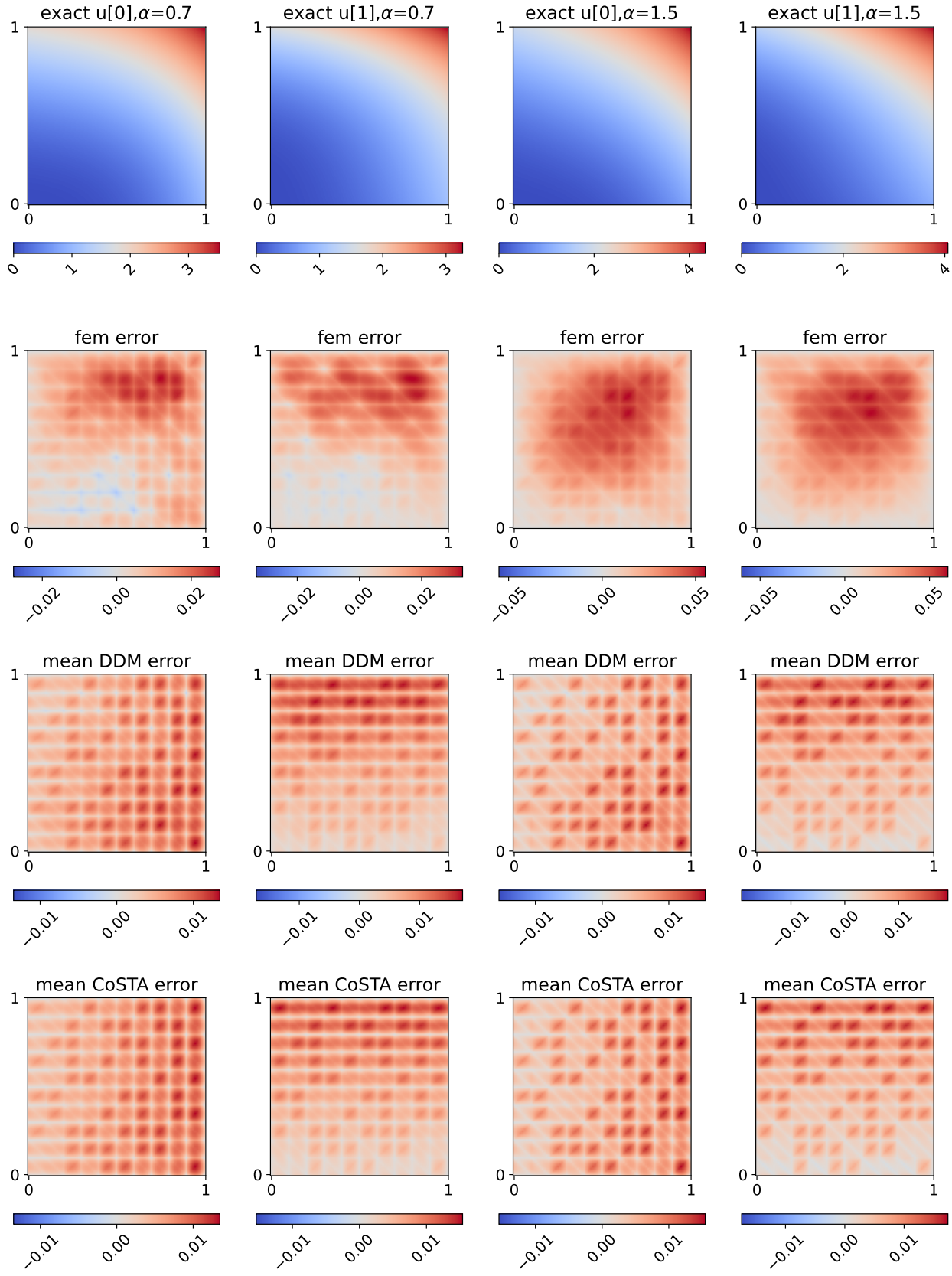


Figure A.4.3: Solution *n3*, interpolation

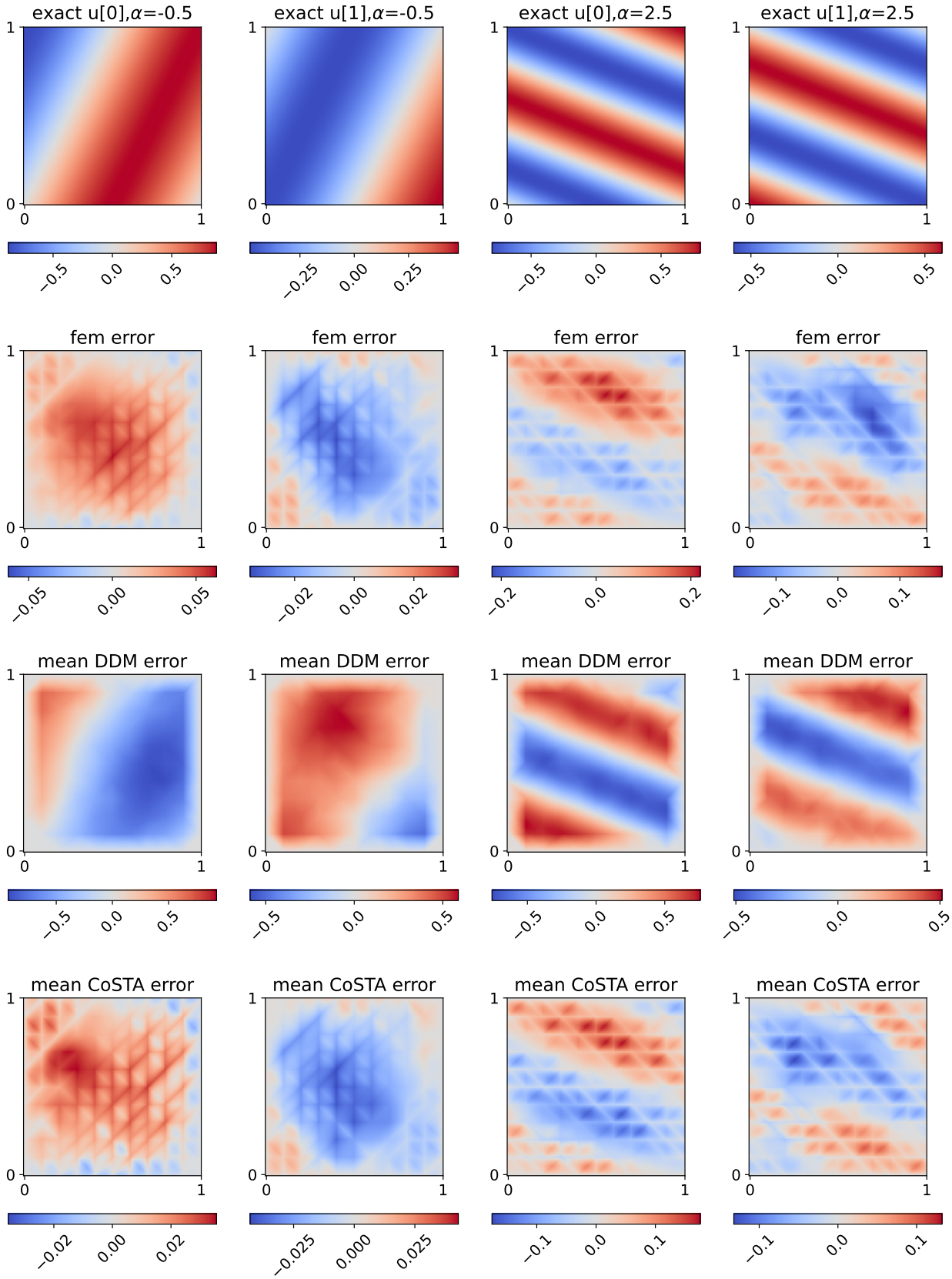


Figure A.4.4: Solution *n1*, extrapolation

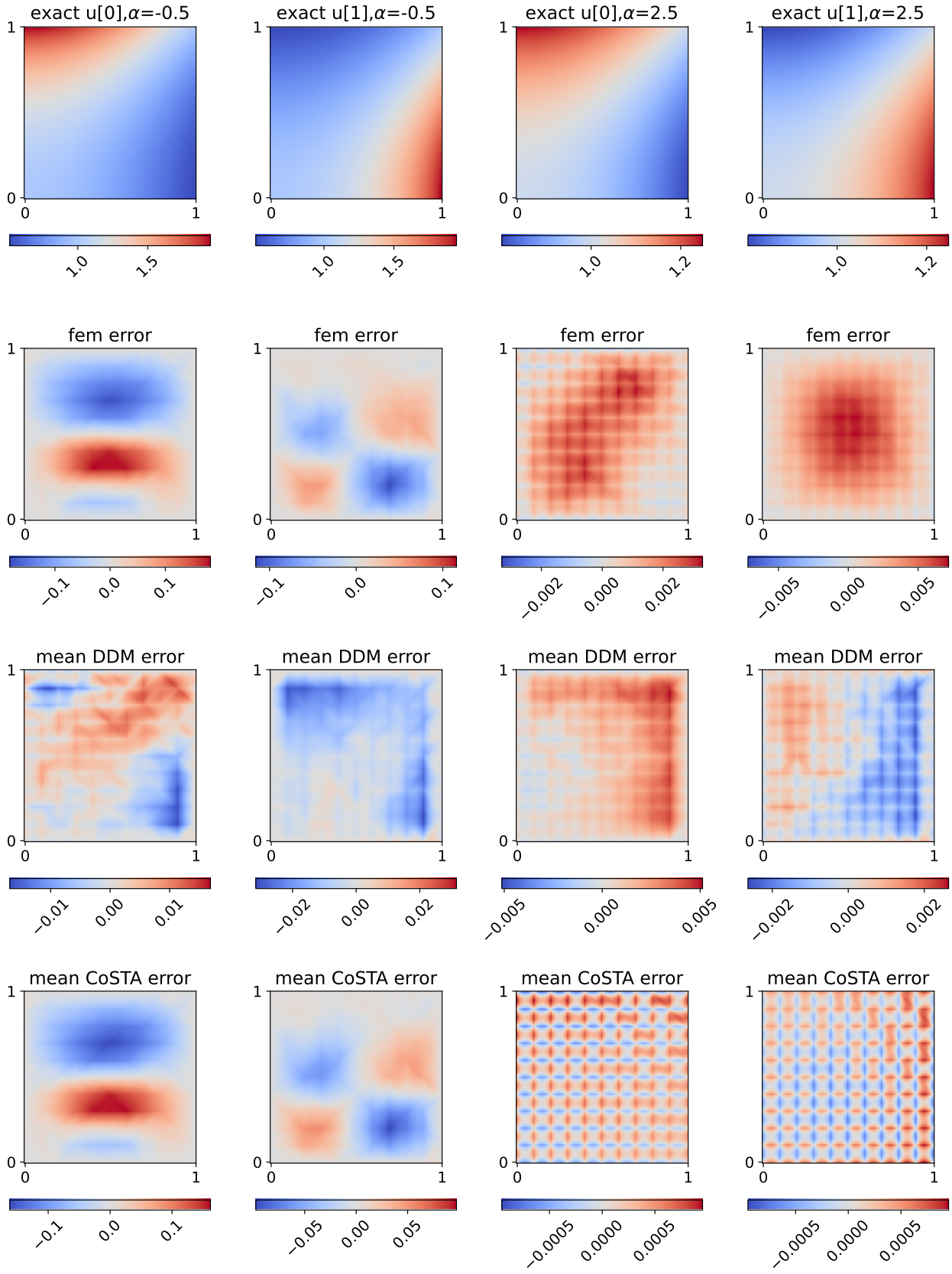


Figure A.4.5: Solution *n2*, extrapolation

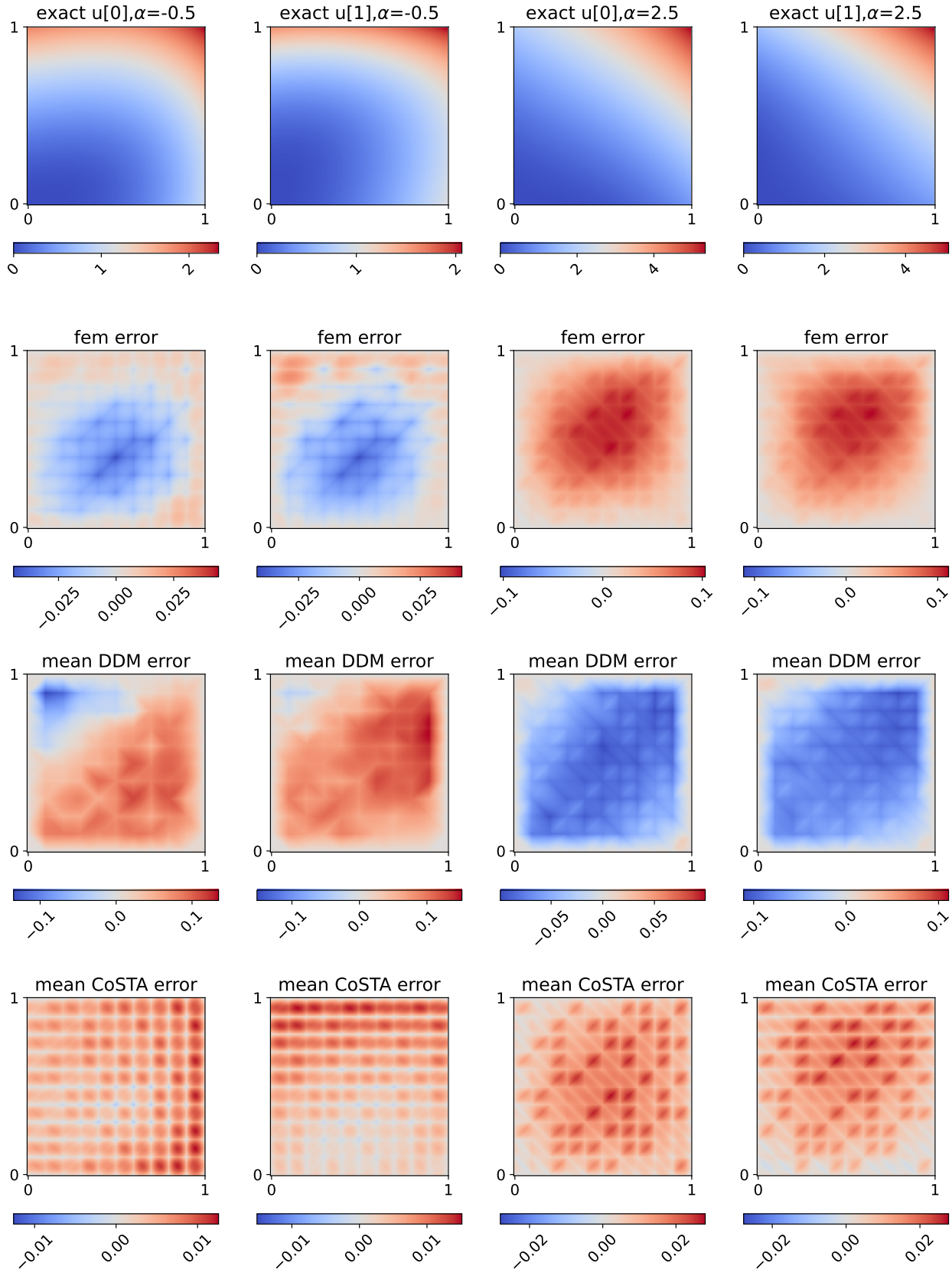
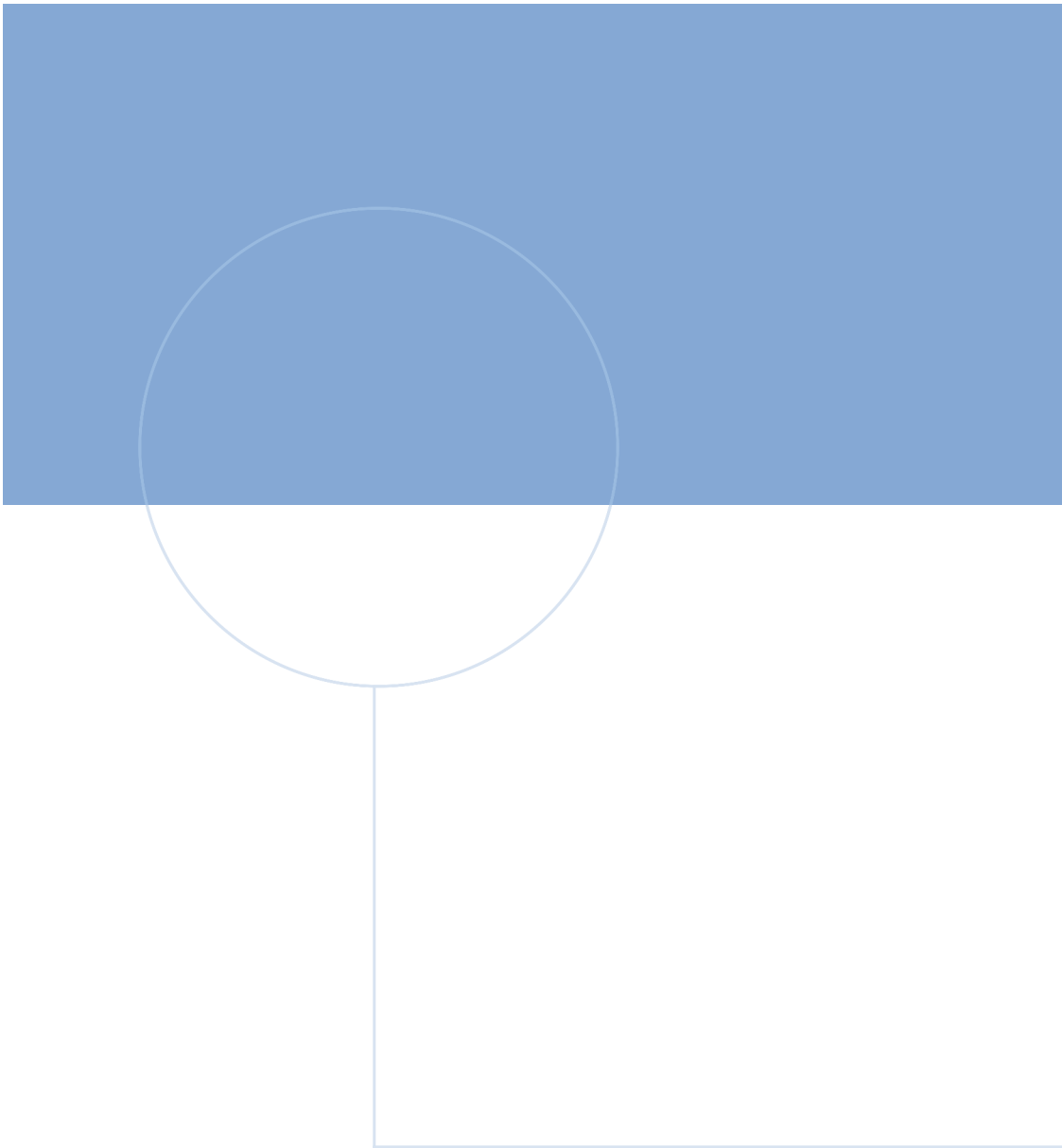


Figure A.4.6: Solution *n3*, extrapolation



NTNU

Norwegian University of
Science and Technology