Fault Tolerant Computation of Hyperbolic Partial Differential Equations with the Sparse Grid Combination Technique

Brendan Harding

August 2015
(revised April 2016)

A thesis submitted for the degree of Doctor of Philosophy
of The Australian National University

Declaration

Some of the work in this thesis has been published jointly with others, see for example [67, 68, 5, 120, 69, 3, 66, 80, 79], however my own contribution to these works has been the development and analysis of the fault tolerant combination technique which is the focus of this thesis. The work in this thesis is my own except where otherwise stated.

Brendan Harding

Acknowledgements

I would like to acknowledge my supervisor Markus Hegland and the other members of my supervisory panel Michael Barnsley and Steve Roberts. I had the privilege of being able to contribute to a collaborative project (project LP110200410 under the Australian Research Council's Linkage Projects funding scheme) and would like to thank the other members of the project for the many interesting discussions. Specifically thanks to Peter Strazdins, Mohsin Ali, Linda Stals, Alistair Rendell and Jay Larson. Fujitsu Laboratories of Europe was the collaborative partner in this project and I would also like to thank Ross Nobes, James Southern and Nick Wilson for the interactions I had with them throughout the project and for hosting me for 6 weeks in the Hayes office in 2014. I was also fortunate to interact with many members of the sparse grid and high dimensional approximation communities. I am very grateful having had the opportunity to share and discuss ideas with the many members of these communities, particularly those who reside in Germany through my short stays in Munich and the many visitors of the Mathematical Sciences Institute. Many computations relating to this work were performed on Raijin, the peak system at the NCI Facility in Canberra, Australia, which is supported by the Australian Commonwealth Government. Finally, a big thank you to my family and friends for their support and encouragement throughout my studies.

Abstract

As the computing power of supercomputers continues to increase exponentially the mean time between failures (MTBF) is decreasing. Checkpoint-restart has historically been the method of choice for recovering from failures. However, such methods become increasingly inefficient as the time required to complete a checkpoint-restart cycle approaches the MTBF. There is therefore a need to explore different ways of making computations fault tolerant. This thesis studies generalisations of the sparse grid combination technique with the goal of developing and analysing a holistic approach to the fault tolerant computation of partial differential equations (PDEs).

Sparse grids allow one to reduce the computational complexity of high dimensional problems with only small loss of accuracy. A drawback is the need to perform computations with a hierarchical basis rather than a traditional nodal basis. We survey classical error estimates for sparse grid interpolation and extend results to functions which are non-zero on the boundary. The combination technique approximates sparse grid solutions via a sum of many coarse approximations which need not be computed with a hierarchical basis. Study of the combination technique often assumes that approximations satisfy an error splitting formula. We adapt classical error splitting results to our slightly different convention of combination level.

Literature on the application of the combination technique to hyperbolic PDEs is scarce, particularly when solved with explicit finite difference methods. We show a particular family of finite difference discretisations for the advection equation solved via the method of lines has solutions which satisfy an error splitting formula. As a consequence, classical error splitting based estimates are readily applied to finite difference solutions of many hyperbolic PDEs. Our analysis also reveals how repeated combinations throughout the computation leads to a reduction in approximation error.

Generalisations of the combination technique are studied and developed at depth. The truncated combination technique is a modification of the classical method used in practical applications and we provide analogues of classical error estimates. Adaptive sparse grids are then studied via a lattice framework. A detailed examination reveals many results regarding combination coefficients and extensions of classical error estimates. The framework is also applied to the study of extrapolation formula. These extensions of the combination technique provide the foundations for the development of the general coefficient problem. Solutions to this problem allow one to combine any collection of coarse approximations on nested grids.

Lastly, we show how the combination technique is made fault tolerant via application of the general coefficient problem. Rather than recompute coarse solutions which fail we instead find new coefficients to combine remaining solutions. This significantly reduces computational overheads in the presence of faults with only small loss of accuracy. The latter is established with a careful study of the expected error for some select cases. We perform numerical experiments by computing combination solutions of the scalar advection equation in a parallel environment with simulated faults. The results support the preceding analysis and show that the overheads are indeed small and a significant improvement over traditional checkpoint-restart methods.

Contents

Chapter 1

Faults in High Performance Computing

In this thesis we are concerned with the development of fault tolerant algorithms for solving PDEs. In order to motivate this we discuss the occurrence of faults in current and future computer systems. Questions of importance are why do faults occur, how often do faults occur, what types of faults occur, and how do faults affect computations? After surveying several articles which attempt to answer these questions in Section 1.1 we start to develop several different models of faults. As faults are random in nature these models are necessarily stochastic. In Section 1.2 fault models for a single processor are discussed. As a high performance computer is essentially collection of electrical components working in parallel this will be a fundamental building block of system models which are considered in Section 1.3. Lastly, in Section 1.4 we apply fault models to some simple calculations in a parallel environment, we review the estimation of the optimal checkpoint frequency for checkpoint-restart based routines and we study the problem of simulating faults.

1.1 Motivation and background

1.1.1 Faults in HPCs

In the 1940s and 1950s the first all-electric computers were constructed using vacuum tubes as the electric switches that performed bitwise operations. The most notable computers of the time consisted of several thousand vacuum tubes, the ENIAC (Electronic Numerical Integrator And Computer) had over 17 000.

There were several issues with vacuum tubes at the time including size, cost, and a relatively short mean time to failure. Now it is important to clarify what we mean by relatively short life span. A long-life vacuum tube at the time had a life expectancy in the order of $\mathcal{O}(5\,000)$ hours [97] (some specially designed tubes in the 1950s had much more) which was reasonable given the manufacturing processes of the time. However, having several thousand such vacuum tubes operating at once it was observed that tubes would fail and need to be replaced at a frequency in the order of days. A notable example is the ENIAC for which it was stated that a tube failed roughly every 2 days [2].

Since the advent of transistors and their incorporation into integrated circuits it has been possible to have large numbers of electric switches on a single component. This soon lead to components in the early 1970s whose life expectancy was of a similar magnitude as the vacuum tube but could do the same amount of work as computers containing thousands of such tubes. This lead to a significant improvement in the reliability of computers vs computing power. With improved photolithography techniques allowing for ever smaller transistors to be etched into semiconductors the trend continued allowing what is now in the order of billions of transistors on a single electrical component whose lifetime exceeds that of the average vacuum tubes in the first all-electric computers. Modern CPUs are so reliable that failures in a modern computer are more likely to have root cause in other components like the power supply, hard drives, cooling systems, etc.

One would therefore expect that reliability is not a big issue for modern day computers. To a large extent this is correct and there have been several decades for which reliability has not been a significant issue for high performance computing, certainly not to the extent of the early vacuum tube computers. However, in recent years it has become increasingly difficult to put more transistors on a single chip. This has caused a slowing of Moore's law (the prediction that the number of transistors on a single chip doubles every two years [100]). Despite this supercomputers have continued to grow at roughly the same rate (although this is starting to slow now it is clearly delayed from the slowing of transistors in CPUs). This has been achieved by increasing parallelism, that is by adding more CPUs to each system. As a result, new systems have significantly more components and now that we have exceeded petascale computing and approach exascale computing there are several computers with cores in the order of millions. This typically means in the order of $\mathcal{O}(100\,000)$ compute nodes each built with components whose life expectancy is in the order of $\mathcal{O}(100\,000)$ hours. It is clear that this is similar to the situation faced by the vacuum tube computers,

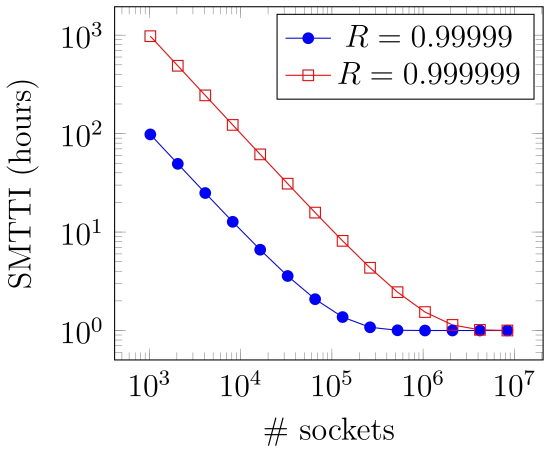


Figure 1.1: *System mean time to interrupt (SMTTI) versus the number of sockets N for different one-hour socket reliabilities $R = 0.999\,99$ and $R = 0.999\,999$ [26]. The plot is of the estimate $SMTTI = 1/(1 - R^N)$ which is flawed in that the limit is 1 hour as $N \to \infty$ with $0 < R < 1$.*

that is where the number of components is similar in magnitude to the expected life time of these components (in hours). Thus we would expect that reliability will again become an issue for high performance computing. Whilst this discussion so far has been somewhat anecdotal, several recent studies and surveys have observed this trend.

In recent years there have been several survey articles predicting a decreasing mean time between failure in high performance computers, see for example [26, 117, 27, 105]. The major contributing factor for this trend is typically identified to be increasing system sizes. As the clock rate of individual cores is no longer increasing in significant amounts, higher performance is achieved primarily by increasing the number of cores. Although this is partially offset by increasing numbers of cores per socket, the largest systems contain an increasing numbers of components. As one would reasonably expect twice as many components to fail twice as often, the increases in number of components is the primary driver of decreasing MTBF (mean time between failures). More precisely, if a single component has a (constant) failure rate F, that is in any given hour it has probability F of failing, then it has probability $1 - F$ of not failing in a given hour. It follows that for N identical components with the same (constant) failure rate the probability that none fail within any given hour is $(1 - F)^N$, thus at least one fails with probability $1 - (1 - F)^N$. For small F (and $N << 1/F$) this may be approximated as $1 - (1 - NF + \mathcal{O}(F^2)) = NF + \mathcal{O}(F^2)$. Figure 1.1

demonstrates the decrease in system mean time to interrupt (SMTTI) for some different levels of reliability $R = 1 - F$ based on this simple model. This trend was observed by Schroeder and Gibson in their study of computers at LANL (Los Alamos National Laboratory) which found that the rate of failure was roughly proportional to the number of nodes in each system [116].

A second contributing factor is that the reliability of individual components is likely to decrease as feature size becomes smaller [33, 28] and chips become more heterogeneous due to increased integration. The reliability of components is also closely linked with energy consumption. For example, typically noise in the CPU is addressed by error detection and correction mechanisms built into the hardware. However this consumes a significant portion of the total energy. As improving energy efficiency is also one of the big challenges for exascale computing [37], it seems unlikely that increased CPU noise caused by decreasing feature size can be entirely addressed by changes in hardware if the energy target of 100 MW for an exascale computer is to be achieved. Compute nodes are also becoming increasingly complex particularly as general purpose graphics processing units and other accelerators become integrated into more systems. Similarly to the entire HPC system, the reliability of individual nodes will likely decrease as more components are added to them.

These arguments would indicate that hardware is the primary source of faults in HPC's, that is that failures in the system occur when a physical component breaks down. Faults are typically classified into one of three categories [101]:

- permanent fault: a component may completely break down such that it no longer functions or produces incorrect results,
- transient fault: a component may temporarily have reduced performance or produce incorrect results,
- intermittent faults: a component oscillates between correct and incorrect operation.

One may also have subcategories where a component produces incorrect results, if these are not detected by the system they are often referred to as silent errors. The frequency at which faults from each category occurs is not well studied, rather studies typically look at the root cause of a faults. Typical causes tabulated include hardware, software, environment and unknown. Such studies reveal that hardware is not the only source of faults in a system and that software faults are also a significant cause. Just how common software faults are compared

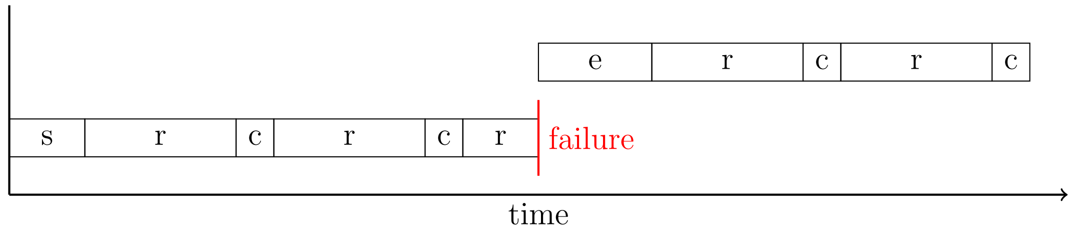


Figure 1.2: *Schematic of a simple checkpoint restart implementation. Block 's' is for program startup, 'r' is main program execution, 'c' is for writing checkpoints, 'e' is the restart of the application and loading of the previous checkpoint.*

to hardware faults is not so clear as different studies in the literature give very different results. The recent study of Schroeder and Gibson showed that hardware is the most common cause followed by software. However a other studies [102, 93, 55] have indicated that software was the largest cause of failures. In this thesis we are not particularly concerned with the root cause but it is clear that more systems need to be studied if we are to better understand the causes and effects of faults in a system.

In most current systems, regardless of the type and origin of a fault, an application using resources which are affected by a fault is interrupted [27]. As such, the affected application must be restarted, either from the beginning or from a previously saved state. The approach of periodically saving the state of the application such that it may be restarted from a relatively recent state is known as checkpoint restart and a simple example is depicted in Figure 1.2. There are a couple of reasons for its success. First, its simplicity means that it typically requires little effort by application developers and system managers to implement and support. Second, the MPI standard specifies that the default error handler for MPI_COMM_WORLD is MPI_ERRORS_ARE_FATAL and as a result MPI implementations typically do not support continued use of MPI after a failure has occurred (`http://www.mpi-forum.org`). As a result of its success, checkpoint restart is the primary solution to fault tolerance implemented on most, if not all, current HPC systems. This presents a problem, as not only is the frequency of faults increasing with system size, the time required to take a global checkpoint also increases with system size. There are two main reasons for this, first some level of synchronisation across the application if the checkpoint is to be consistent, and second increased data size takes longer to write to stable storage due to limitations in bandwidth. In particular, if the MTBF is less then the time required to complete a checkpoint restart cycle (i.e. time to restart from a checkpoint and then immediately save a checkpoint) then machine utilisation will be extremely low [117, 26, 117, 40]. Significant work has been put into improving the checkpoint restart approach including the use of uncoordinated checkpoints [6, 41], non-blocking checkpoints [29, 96], diskless checkpointing [106, 107, 108, 42] and message logging [87, 14, 15]. However, it is imperative that alternative approaches to fault tolerance are also developed and evaluated.

Several other categories of approaches to fault tolerance are discussed in the literature, see for example the surveys [26, 27]. We have already discussed checkpoint restart based approaches, sometimes also referred to as rollback recovery. Replication is often discussed as a viable option if the utilisation of other approached drops below 50%. At this threshold the cost of duplicating computations is competitive and can tolerate faults affecting one of the two duplicates. Failure prediction is another approach where by continuous analysis of the system is used to predict when particular components may be about to fail [49]. Computations can then be moved onto other resources in order to avoid failures. Another concept that has been recently proposed is selective reliability where parts of an algorithm which are naturally fault tolerant can be run on less reliable hardware at a lower energy cost whilst critical parts are run on highly reliable (and more expensive) hardware [16]. The implication is that the use of naturally fault tolerant algorithms may also provide solutions to the energy problem. Algorithm based fault tolerance (ABFT) is typically based on a fail-continue model in which a process fails but continues operation possibly providing incorrect results. In some cases such errors are detected by the system and corrected, in others they go undetected and are thus referred to as silent. Most ABFT deals with the detection and correction of these silent errors which we review in the next subsection.

1.1.2 Algorithm based fault tolerance

ABFT is typically referenced as beginning with the work of Huang and Abraham on detecting and correcting silent errors in matrix-matrix multiplication [74]. However, naturally fault tolerant algorithms, a subset of ABFT of algorithms which are self correcting such as those based on optimisation [118], can be traced back much further. Gauss made the comment that iterative methods were error tolerant [51][1], of course he was not referring to computer errors but rather human errors. If an arithmetic error is made on one iteration it did not matter as subsequent iterations would still converge. Nonetheless, Huang and Abraham

[1]See Gauss' letter to his student Gerling on pp.278–281.

opened up an entire area of research devoted to error detection and correction in a large range of matrix based calculations.

Let $A, B \in \mathbb{R}^{n \times n}$ be real matrices and $e = (1, 1, \dots, 1) \in \mathbb{R}^n$ a vector and consider the computation of $C = AB$. Huang and Abraham observed that adding a checksum row to A and a checksum column to B led to the computation

$$\begin{bmatrix} A \\ e^T A \end{bmatrix} [B \ Be] = \begin{bmatrix} AB & A(Be) \\ (e^T A)B & (e^T A)(Be) \end{bmatrix} = \begin{bmatrix} C & Ce \\ e^T C & e^T Ce \end{bmatrix} .$$

Assuming that $(e^T A)B$ and $A(Be)$ are correctly computed then errors in C can be detected by comparing the vectors $(e^T A)B, e^T C$ and $A(Be), Ce$. If a single silent error occurred for a given C_{ij} then one would find $(e^T AB)_j \neq (e^T C)_j$ and $(ABe)_i \neq (Ce)_i$ thus giving the location of the error and further one can correct the value via

$$C_{ij} = (e^T AB)_j - \sum_{k \neq i} C_{k,j} = (ABe)_i - \sum_{k \neq j} C_{i,k} .$$

Multiple errors can be detected and corrected in a similar way as long as the location of errors in the matrix allowed the locations to be uniquely identified by the errors in the checksums. They demonstrate empirically that the proportion of these exceptional occurrences decreases as n increases when location of errors are randomised.

The computation $C = AB$ consisted of n^2 dot products of length n vectors whilst the computation with checksums consists of $(n + 1)^2 + 2n$ dot products of length n vectors, where the additional $2n$ is for the computation of $e^T A$ and Be. Thus the relative overhead is $\frac{4n+1}{n^2} = \mathcal{O}(1/n)$. For parallel computation they developed several partitioned checksums which could be used for detection and recovery in a similar fashion. In summary their recovery algorithm exhibits greater coverage (i.e. can recover from a larger proportion of errors) and less overhead as the problem size increases. As a result one would expect the approach to scale extremely well to large problems on large machines. This is a staggering result contrasting the decreasing performance of checkpoint restart based approaches with increasing system and problem sizes.

Since the original paper was published there has been a significant amount of work on the analysis and extension ABFT [8, 94, 114]. More recently there has been significant work on matrix factorisations (including Gaussian elimination) and implementation within scientific software [31, 13, 38, 128, 124]. Most of this work continues to focus on the use checksums or other matrix encoding/decoding schemes which can be checked throughout a computation. Algorithm based fault tolerance has also been considered in other problems like Newton's method [90], heat transfer [92], iterative methods [30], stochastic PDEs [103] and many others. Naturally fault tolerant algorithms have also received some attention in the literature [52, 118]. Whilst work on fault tolerance is slowly broadening, the majority of the literature surrounds the detection and correction of silent errors within linear algebra computations. A significant difficulty in the research of fault tolerant algorithms is a lack of support in the MPI standard for detecting process errors and reconstructing communicators so that an application can continue. There have been a few efforts in implementing such support within MPI most notably FT-MPI [43, 44, 46, 45, 47] and more recently User Level Fault Mitigation (ULFM) [11, 10]. The former was able to survive the failure of $n - 1$ processes in a n process job and respawn them. Unfortunately it was built on the MPI 1.2 specification which is now outdated. The latter shows some promise and some effort has been made to get their work accepted into the MPI 3 standard. However at this time it is not a part of the standard and the implementation is in a beta phase.

There seem to be two knowledge gaps in the literature. Although some work has been done on heat transfer and iterative methods it is not clear how these methods will apply more generally to time evolving PDEs, particularly those based on explicit methods. Second, the ABFT in the literature is typically not designed to cope with fail-stop faults, that is where a fail results in loss of all data on one or more processors. Thus in practice it may be necessary to use checkpointing alongside ABFT if all bases are to be covered. Another observation to be made is that much of the effort has been focused on exact recovery of errors and/or lost data in both checkpointing and ABFT research. (An exception is the work done in the context of stochastic PDEs [103], but we refer mainly to algorithms which are not stochastic in nature). Whilst this is a sensible goal in terms of repeatability of computations it contrasts the approach used in many other systems, telecommunication for example, where a temporary performance degradation is typically tolerated and even preferred over complete loss of functionality. Given the energy challenges facing exascale computing and the expense of requiring exact recovery in all circumstances it may be sensible to consider algorithms which allow computations to continue through faults but producing (slightly) degraded results. One might call upon the HPC community to be more tolerant of faults whilst conducting research on fault tolerance.

In this thesis we aim to address these gaps by the development and analysis of a new form of algorithm based fault tolerance based on the sparse grid combination technique (introduced in Chapter 2). At the same time we hope to suggest a new paradigm for fault tolerant computing in which one is able to trade off recovery times for slightly degraded solutions. To reiterate, our approach differs from existing work in several ways. First, it is a much more holistic approach with respect to making the computation fault tolerant. The approach can be used to survive a wide variety of faults from fail-stop faults to silent errors (coupled with a suitable detection algorithm). Additionally, rather than focus on one aspect of the computation like the linear algebra, the majority of the computation is made fault tolerant by this approach. Second, the approach is applicable to a wide variety of PDEs for which many of the previously developed ABFT algorithms are not applicable. Third, the ability to trade off increased recovery time for slightly degraded solution (in algorithms which are not stochastic in nature) is something new to the HPC fault tolerance literature and is something we feel is worth investigating.

This thesis is organised as follows. The remainder of this chapter is devoted to the development of fault models which we use later in the analysis of the proposed fault tolerant combination technique (FTCT). In Chapter 2 we introduce sparse grids and the combination technique. We review the classical error analysis as the techniques used here will be extended to the study of the FTCT. In Chapter 3 we review some simple hyperbolic PDEs. We describe the problems that are used for numerical results throughout the thesis and discuss how the combination technique is applied to such problems. In Chapter 4 some extensions and generalisations of the combination technique are developed. The main contribution is a detailed analysis of adaptive sparse grids and extension of some of this work to extrapolation techniques. A generalised combination technique for the combination of arbitrary collections of approximations on regular grids is also developed from this work. This leads naturally to a fault tolerant algorithm. In Chapter 5 we perform numerical analysis on the FTCT. We compute the expected error of some specific applications of the FTCT using simplified fault models. An analysis of the computation overhead is also given.

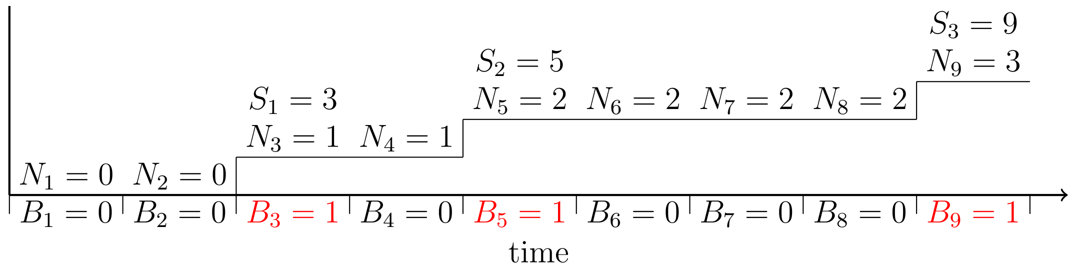
1.2 Component fault models

In this section we look at two stochastic models of the state of individual components of a machine, specifically a central processing unit (CPU). We first consider sampling the state of a processor at regular intervals which will be modelled as a sequence of Bernoulli trials. Following this, renewal processes are reviewed as a model for the number of failures that occur over time for a processor which is replaced upon failure. These simple models will form the building blocks for modelling failures in machines consisting of many processors which operate in parallel in Section 1.3. Note that the models discussed in this section are not specific to computer processors and could be equally applied to any component (electrical or mechanical) which has finite expected lifetime.

1.2.1 Bernoulli trials

In the simplest of circumstances we can consider a processor as being in one of two states at any given time. Either it is 'operating as intended' or it is 'not operating as intended'. In the state 'not operating as intended' we include the possibilities that the processor produces no output at all or produces incorrect output. To simplify the discussion we refer to these two states of operation as 'on' and 'failed' respectively. The state of a processor is then observed at regular intervals of length t. It is assumed that data computed in the previous interval is collected during each observation. If the processor if found to be in the failed state then all computations from the preceding interval are lost and we may therefore consider the process as being in the failed state throughout the entire interval. When a processor is observed in the failed state it is instantly replaced with another processor which is statistically identicals with respect to operating characteristics. A reasonable first failure model is to simply keep track of the proportion of times the processor was found to be in the failed state versus the total number of observations. After a sufficiently large number of observations the proportion of observations in the failed state can be used as an approximation of the probability of subsequent observations being in the failed state. This experimental setup can be modelled as a sequence of Bernoulli trials.

More formally, let $B_1, B_2, B_3, \dots$ be a sequence of independent and identically distributed random variables for which each B_i denotes the state of the processor throughout the ith interval. Let $B_i = 0$ denote the state 'on' and $B_i = 1$ denote the state 'failed' in the ith interval occurring probability $1 - p$ and p



$S_3 = 9$
$N_9 = 3$
$S_2 = 5$
$N_5 = 2$ $N_6 = 2$ $N_7 = 2$ $N_8 = 2$
$S_1 = 3$
$N_3 = 1$ $N_4 = 1$
$N_1 = 0$ $N_2 = 0$
$B_1 = 0$ $B_2 = 0$ $B_3 = 1$ $B_4 = 0$ $B_5 = 1$ $B_6 = 0$ $B_7 = 0$ $B_8 = 0$ $B_9 = 1$
time

Figure 1.3: *The Bernoulli trial model of process failure. B_i is the status of the ith interval with 1 being a failure. The N_i are the cumulative sum of the B_i. S_j denotes the first i for which $N_i \geq j$.*

for some $p \in (0, 1)$ respectively, that is $B_i \sim B(1, p)$ (where $B(m, q)$ denotes the binomial distribution having probability mass function $\binom{m}{k} q^k (1-q)^{m-k}$ with $k \in \{0, 1, \ldots, m\}$). It is implicit in this model that failure rate is constant over time. Checking the processor at the end of the ith interval is then synonymous with sampling the Bernoulli random variable B_i, also known as a Bernoulli trial. We note that $\mathrm{E}[B_i] = (1-p) \cdot 0 + p \cdot 1 = p$ which is the proportion of intervals for which the processor can be expected to be in the failed state. Additionally the variance is $\mathrm{E}[B_i^2] - \mathrm{E}[B_i]^2 = p - p^2 = p(1-p)$. Given this simple model there are many questions one might ask:

- How many replacement processes are required over n intervals?
- What is the downtime and availability over n intervals?
- What is the time to the first failure?
- What is the expected lifetime of each processor?

The number of replacement processors required over n intervals is given by the random variable $N_n = \sum_{i=1}^{n} B_i$ (which also counts the number of failures). It follows that the number of replacements has expectation $\mathrm{E}[N_n] = np$ and variance $\mathrm{Var}(N_n) = np(1-p)$. The proportion of failures in n checks is given by N_n/n. By the strong law of large numbers

$$\frac{N_n}{n} \xrightarrow[n\to\infty]{\text{a.s.}} E[B_1] = p\,,$$

(with $X_n \xrightarrow{\text{a.s.}} x$ meaning $\Pr(X_n \to x) = 1$). Additionally

$$\mathrm{E}\left[\frac{N_n}{n}\right] = \frac{\mathrm{E}[B_1 + \cdots + \mathrm{E}[B_n]]}{n} = \frac{np}{n} = p\,.$$

The total downtime over n intervals is given by tN_n with expectation $\mathrm{E}[tN_n] = tnp$ and variance $\mathrm{Var}(tN_n) = t^2np(1-p)$. The availability is given by $t(n-N_n)$ which has expectation $tn(1-p)$ and variance $t^2np(1-p)$. The time to first failure is tS_1 where $S_j := \min_i \{N_i \geq j\}$ for $j = 1, 2, 3, \ldots$ which denotes the number of intervals before the jth failure. Figure 1.3 depicts the Bernoulli trial model with the random variables B_i, N_i and S_j. As $S_1 = k$ iff $B_i = 0$ for $i = 1, 2, \ldots, k-1$ and $B_k = 1$ it follows that

$$\mathrm{E}[S_1] = \sum_{k=1}^{\infty} kp(1-p)^{k-1} = \frac{p}{1-p} \sum_{k=1}^{\infty} \sum_{i=k}^{\infty} (1-p)^i = \frac{p}{1-p} \sum_{k=1}^{\infty} \frac{(1-p)^k}{1-(1-p)} = \frac{1}{p}\,.$$

Therefore the expected time to the first failure is $\mathrm{E}[tS_1] = \frac{t}{p}$. The time between failures is given by $S_{j+1} - S_j$ but since the rate of failure is constant the expected time between failures is equal to the expected time to the first failure. It follows that the expected lifetime of each processor is $t\mathrm{E}[S_{j+1} - S_j] = \frac{t}{p}$.

Whilst this is a very simple model there are a variety of cases where limited information about failure rates is available and this model has practical use. Suppose you have bought a processor and switch it on. The only information you may have available regarding how long that component will operate successfully may be the manufacturers rated lifetime. Suppose the processor is rated for M continuous operating hours, then if we take M to be the expected lifetime then our Bernoulli model says $\mathrm{E}[tS_1] = \frac{t}{p} = M$ and thus $p = \frac{t}{M}$ is the probability of failure for each observation (with t the time between observations in hours). As a systems administrator you may be expected to ensure a processor is available for use for 4 years (35 064 hours). The number of observations over these 4 years is $\frac{35064}{t}$. It follows that the number of replacements has expectation $\mathrm{E}[N_n] = np = \frac{35064}{t}\frac{t}{M} = \frac{35064}{M}$.

In Section 1.2.2 we will consider a renewal process model of faults. To motivate this we will consider the failure distribution the Bernoulli trial model where the length of the interval t between observations vanishes. We are interested in a total time s for which there are $n = \lceil s/t \rceil$ intervals. In the infinitesimal limit $t \to 0$ we assume the probability of failure in each interval is proportional to t. Specifically, let p_t be the probability of failure within an interval of length t and $\lim_{t\to 0} \frac{p_t}{t} = r < \infty$. We now claim that as we increase the frequency of observation, that is $t \to 0$, then the probability of observing k failures in a fixed interval, that is $\Pr(N_n = k)$, converges to a Poisson distribution. The probability of observing k failures out of the n observations is given by the binomial distribution $N_n \sim B(n, p_t)$ having probability mass function

$$\Pr(N_n = k) = \binom{n}{k} p_t^k (1-p_t)^{n-k} \,.$$

We may rewrite this as

$$\Pr(N_n = k) = \frac{1}{k!}(np_t)((n-1)p_t)\cdots((n-k+1)p_t)\left(1-\frac{np_t}{n}\right)^{n-k} \,.$$

Now as $n \to \infty$ (and $t = s/n \to 0$) one has

$$np_t = \frac{s}{t}p_t \to sr \,.$$

Similarly $(n-1)p_t \to sr, \ldots, (n-k+1)p_t \to sr$ for fixed k. This leads to the limit

$$\begin{aligned}\left(1-\frac{np_t}{n}\right)^{n-k} = \left(1-\frac{np_t}{n}\right)^{-k}\left(1-\frac{np_t}{n}\right)^{n} &\xrightarrow{t\to 0} \left(1-\frac{sr}{n}\right)^{-k}\left(1-\frac{sr}{n}\right)^{n} \\ &\xrightarrow{n\to\infty} 1 \times e^{-sr} \,.\end{aligned}$$

Putting the pieces together one obtains the limit

$$\binom{n}{k} p_t^k (1-p_t)^{n-k} \xrightarrow[t\to 0]{} \frac{(sr)^k}{k!}e^{-sr} \,,$$

which is the Poisson distribution. Thus N_n converges to a Poisson process which is a special example of a renewal process which is introduced in the following section.

1.2.2 Renewal processes

Definition 1.1. Let $X_1, X_2, X_3, \ldots$ be a sequence of independent and identically distributed random variables with support $[0, \infty)$ and (strictly) positive and finite expectation (i.e. $0 < \mathrm{E}[X_i] < \infty$). Now for $i = 1, 2, 3, \ldots$ we define the sequence of random variables $S_i := \sum_{j=1}^{i} X_j$. Associated with the S_i we have for $t \geq 0$ the counting process

$$N(t) := \sup\{i : S_i \leq t\} = \sum_{i=1}^{\infty} \chi_{[0,t]}(S_i) \,.$$

which is called a renewal process.

Renewal processes are well known in the study of failures/reliability, queues, arrival of messages (e.g. email or phone calls), survival and numerous other

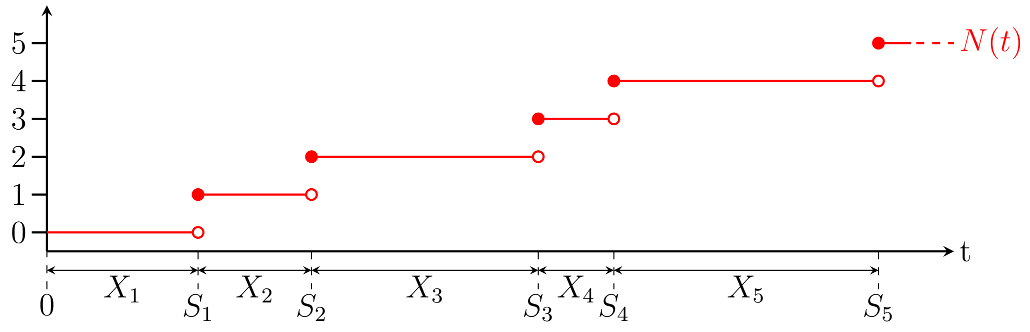


Figure 1.4: *The renewal model of process failure. The X_i are the times between failures and the S_i are the cumulative sum of the time between failures. $N(t)$ is the renewal process and is the number of failures that have occurred up to time t.*

examples. In the context of processor faults we think of the X_i as denoting the time between the $i-1$ and ith failures of a processor (which is immediately replaced with an identical processor after each such failure). Then S_i is the time at which the ith processor fails and $N(t)$ is the number of failures that have occurred up to (and including) a given time t. An example of a renewal process is depicted in Figure 1.4.

Example 1.2. Suppose we have X_i for which the probability of failing at any instant of time is 0 and the probability of failure within any interval is independent of when the interval begins. We claim that the X_i must be exponentially distributed and the corresponding renewal process $N(t)$ is a Poisson process. More formally let $X_1, X_2, \dots$ be continuous and $\Pr(X_i \leq t+s \mid X_i > t) = \Pr(X_i < s)$ for $t, s \geq 0$. We will denote the cumulative distribution of the X_i by $F(t) = \Pr(X_i \leq t)$. For all $t, s \geq 0$ we have

$$F(s) = \frac{F(t+s)-F(t)}{1-F(t)} \implies 1-F(s) = 1 - \frac{F(t+s)-F(t)}{1-F(t)} = \frac{1-F(t+s)}{1-F(t)}$$
$$\implies (1-F(s))(1-F(t)) = 1-F(t+s)\,.$$

It follows that $1-F\left(\sum_{i=1}^n t_i\right) = \prod_{i=1}^n (1-F(t_i))$ for $t_1, \dots t_n \geq 0$. Let $r = 1-F(1)$, then for all $n = 1, 2, \dots$ one has $1-F(n) = 1-F\left(\sum_{i=1}^n 1\right) = (1-F(1))^n = r^n$. Similarly $r^{1/n} = (1-F(1))^{1/n} = \left(1-F\left(\sum_{i=1}^n 1/n\right)\right)^{1/n} = 1-F(1/n)$. Thus for any convergent sequence of (positive) rationals $\frac{m_i}{n_i}$ one has $\lim_{i\to\infty} 1-F(m_i/n_i) = \lim_{i\to\infty} r^{m_i/n_i}$ and since F is continuous it follows that $1-F(t) = r^t$ for all $t \in [0,\infty)$. Thus $F(t) = 1-e^{t\log(r)}$, that is X_i is exponentially distributed with mean $\frac{1}{\log(r)}$ (note that $r = 1-F(1) \in (0,1)$ and so $\log(r) < 0$). Now consider the corresponding renewal process $N(t)$ for a fixed time t. With $S_i = \sum_{j=1}^i X_j$

and F_{S_i} the cumulative distribution function of S_i note that via the law of total probability

$$\begin{aligned}\Pr(N(t)=k)=\Pr(S_k \leq t < S_{k+1}) &= \int_0^t \Pr(X_{k+1} \geq t - S_k \mid S_k = s)\, dF_{S_k}(s) \\ &= \int_0^t \Pr(X_{k+1} \geq t-s) f_{S_k}(s)\, ds\,.\end{aligned}$$

Now $\Pr(X_{k+1} \geq t-s) = e^{(t-s)\log(r)}$ and $f_{S_k(t)} = \frac{dF_{S_k}(t)}{dt}$ is given by the k-fold convolution of the probability distribution $f(t) = \frac{dF(t)}{dt} = -\log(r)e^{t\log(r)}$ of each of the X_i. Assume that $f_{S_k}(s) = \frac{s^{k-1}(-\log(r))^k}{(k-1)!}e^{s\log(r)}$ (which is clearly true for $f_{S_1} = f_{X_1} = f$) then

$$\begin{aligned}f_{S_{k+1}}(s) &= \int_0^s f_{S_k}(s-t)\, dF(t) \\ &= \int_0^s \left(\frac{(s-t)^{k-1}(-\log(r))^k}{(k-1)!}e^{(s-t)\log(r)}\right)\left(-\log(r)e^{t\log(r)}\right)\, dt \\ &= (-\log(r))^{k+1}e^{s\log(r)}\int_0^s \frac{(s-t)^{k-1}}{(k-1)!}\, dt \\ &= (-\log(r))^{k+1}e^{s\log(r)}\left[\frac{-(s-t)^k}{k!}\right]_{t=0}^{s} = \frac{s^k(-\log(r))^{k+1}}{k!}e^{s\log(r)}\,.\end{aligned}$$

Thus by induction $f_{S_k}(s) = \frac{s^{k-1}(-\log(r))^k}{(k-1)!}e^{s\log(r)}$ for all $k = 1, 2, \ldots$ and therefore

$$\Pr(N(t)=k) = \int_0^t \frac{s^{k-1}(-\log(r))^k}{(k-1)!}e^{t\log(r)}\, ds = \frac{t^k(-\log(r))^k}{k!}e^{t\log(r)}\,.$$

Hence $N(t)$ is Poisson distributed. This completes the example.

One may wish to know the average rate at which $N(t)$ grows. As $N(t)$ is constant except at the points $S_1, S_2, \ldots$ where it is discontinuous it follows that $\frac{dN(t)}{dt} = 0$ almost everywhere. This is not a particularly useful result so we instead study $\frac{N(t)}{t}$ for large t.

Lemma 1.3 ([99, 76])**.** *Given a renewal process $N(t)$ with inter-arrival times $X_1, X_2, \ldots$ which are independent and identically distributed (*IID*) one has*

$$\lim_{t\to\infty}\frac{N(t)}{t} = \frac{1}{\mathrm{E}[X_1]}$$

almost surely (that is $\Pr(\lim_{t\to\infty}\frac{N(t)}{t} = \frac{1}{\mathrm{E}[X_1]}) = 1$*).*

Proof. With $S_i = \sum_{j=1}^{i} X_j$ note that $S_{N(t)} \leq t < S_{N(t)+1}$ and therefore

$$\frac{S_{N(t)}}{N(t)} \leq \frac{t}{N(t)} < \frac{S_{N(t)+1}}{N(t)}$$

By the strong law of large numbers we have that

$$\lim_{i\to\infty} \frac{1}{i} S_i = \lim_{i\to\infty} \frac{1}{i} \sum_{j=1}^{i} X_j \xrightarrow{\text{a.s.}} \mathrm{E}[X_1] \,.$$

Similarly

$$\lim_{i\to\infty} \frac{1}{i} S_{i+1} = \lim_{i\to\infty} \frac{i+1}{i} \frac{S_{i+1}}{i+1} \xrightarrow{\text{a.s.}} \mathrm{E}[X_1] \,.$$

Therefore given any sequence of times t_i such that $t_i \to \infty$ and $N(t_i) = i$ one has

$$\frac{t_i}{N(t_i)} < \frac{S_{N(t_i)+1}}{N(t_i)} = \frac{S_{i+1}}{i} \xrightarrow[i\to\infty]{\text{a.s.}} \mathrm{E}[X_1] \,,$$

and

$$\frac{t_i}{N(t_i)} \geq \frac{S_{N(t_i)}}{N(t_i)} = \frac{S_i}{i} \xrightarrow[i\to\infty]{\text{a.s.}} \mathrm{E}[X_1] \,.$$

It follows that $\frac{t_i}{N(t_i)} \xrightarrow[i\to\infty]{\text{a.s.}} \mathrm{E}[X_1]$, or equivalently $N(t_i)/t_i \xrightarrow[i\to\infty]{\text{a.s.}} 1/E[X_1]$. (Note that the existence of the sequence $t_i \to \infty$ with $N(t_i) = i$ for all i is guaranteed by the condition $0 < \mathrm{E}[X_i] < \infty$). $\square$

Therefore for sufficiently large t one has $N(t) \approx \frac{t}{\mathrm{E}[X_1]}$. One might conjecture that $E[N(t)]/t \to 1/E[X_1]$ is an immediate consequence. Whilst the conjecture is true the proof requires some care and the result is referred to as the elementary renewal theorem.

Theorem 1.4 (Elementary renewal theorem [99]). *Let $\{N(t); t \geq 0\}$ be a renewal process with mean inter-arrival time* $\mathrm{E}[X_1]$*, then*

$$\lim_{t\to\infty} \frac{\mathrm{E}[N(t)]}{t} = \frac{1}{\mathrm{E}[X_1]} \,.$$

Several different proofs of this may be found in elementary texts on stochastic processes, see for example [76, 121, 113] or Cox's monograph [35]. We include a proof here that uses the notion of stopping times similar to that in [99].

Definition 1.5. A *stopping time* T for a sequence of random variables $X_1, X_2, \ldots$ is a positive integer valued random variable with $\mathrm{E}[T] < \infty$ for which the event $\{T \geq n\}$ is statistically independent of $X_n, X_{n+1}, \ldots$ (i.e. it may only depend on $X_1, \ldots, X_{n-1}$).

Notice that $N(t)+1$ is a stopping time since the event $\{N(t)+1 \geq n\} = \{S_{n-1} \leq t\}$ depends only upon $X_1, \dots, X_{n-1}$. Clearly $N(t)$ is not a stopping time because $\{N(t) \geq n\} = \{S_n \leq t\}$ depends on X_n. The following theorem regarding stopping times is extremely useful.

Theorem 1.6 (Wald's equation [99]). *Let $X_1, X_2, \dots$ be a sequence of* IID *random variables each with mean $\mathrm{E}[X_1]$. If T is a stopping time for $X_1, X_2, \dots$, and $\mathrm{E}[T] < \infty$, and $S_T = \sum_{i=1}^{T} X_i$, then*

$$\mathrm{E}[S_T] = \mathrm{E}[X_1]\mathrm{E}[T]\,.$$

Proof. We can write

$$S_T = \sum_{i=1}^{\infty} X_i \chi_{[i,\infty)}(T)\,.$$

As the event $T \geq i$ is independent of X_i

$$\begin{aligned}\mathrm{E}[S_T] = E\left[\sum_{i=1}^{\infty} X_i \chi_{[i,\infty)}(T)\right] &= \sum_{i=1}^{\infty} \mathrm{E}[X_i \chi_{[i,\infty)}(T)] \\ &= \sum_{i=1}^{\infty} \mathrm{E}[X_1]\mathrm{E}[\chi_{[i,\infty)}(T)] \\ &= \mathrm{E}[X_1]\sum_{i=1}^{\infty} \Pr(T \geq i) = \mathrm{E}[X_1]\mathrm{E}[T]\end{aligned}$$

where the interchanging of the expectation of infinite sum is valid because $\mathrm{E}[T]$ is finite and the last equality uses the identity

$$\begin{aligned}\mathrm{E}[T] = \sum_{j=1}^{\infty} j \Pr(T=j) = \sum_{j=1}^{\infty}\sum_{i=1}^{j} \Pr(T=j) &= \sum_{i=1}^{\infty}\sum_{j=i}^{\infty} \Pr(T=j) \\ &= \sum_{i=1}^{\infty} \Pr(T \geq i)\,.\end{aligned}$$

□

We now prove the elementary renewal theorem.

Proof of Theorem 1.4. Fix $s > 0$, then for $i = 1, 2, \dots$ we define the truncated random variables $\tilde{X}_i = X_i$ for $X_i \leq s$ and $\tilde{X}_i = s$ for $X_i > s$. As the X_i are IID then the $\tilde{X}_i$ are also IID and by considering $\tilde{S}_i = \sum_{j=1}^{i} \tilde{X}_j$ we form the renewal process $\tilde{N}(t) = \sup\{i : \tilde{S}_i \leq t\}$ for $t \geq 0$. Clearly $\tilde{S}_i \leq S_i$ implies $\tilde{N}(t) \geq N(t)$ and therefore $\mathrm{E}[\tilde{N}(t)] \geq \mathrm{E}[N(t)]$. By Wald's equality

$$\mathrm{E}[S_{N(t)+1}] = \mathrm{E}[X_1]\mathrm{E}[N(t)+1] = \mathrm{E}[X_1](\mathrm{E}[N(t)]+1)$$

and therefore since $S_{N(t)+1} \geq t$ one has

$$\frac{\mathrm{E}[N(t)]}{t} = \frac{\mathrm{E}[S_{N(t)+1}]/\mathrm{E}[X_1] - 1}{t} \geq \frac{1}{\mathrm{E}[X_1]} - \frac{1}{t} \,. \tag{1.1}$$

Similarly for the $\tilde{S}_i$ we consider the stopping time $\tilde{N}(t) + 1$ for which one has $\tilde{S}_{\tilde{N}(t)} \leq t$ and as $\tilde{X}_{N(t)+1} \leq s$ it follows that $\tilde{S}_{\tilde{N}(t)+1} \leq t + s$. Again by applying Wald's equality one obtains

$$\mathrm{E}[\tilde{X}_1](\mathrm{E}[N(t)] + 1) \leq \mathrm{E}[\tilde{X}_1](\mathrm{E}[\tilde{N}(t)] + 1) = \mathrm{E}[\tilde{S}_{\tilde{N}(t)+1}] \leq t + s \,.$$

Re-arranging and dividing by t we have

$$\frac{\mathrm{E}[N(t)]}{t} \leq \frac{(t+s)/\mathrm{E}[\tilde{X}_1] - 1}{t} = \frac{1}{\mathrm{E}[\tilde{X}_1]} + \frac{s}{t\mathrm{E}[\tilde{X}_1]} - \frac{1}{t} \,. \tag{1.2}$$

Now by setting $s = \sqrt{t}$ and combining equations (1.1) and (1.2) one has

$$\frac{1}{\mathrm{E}[X_1]} - \frac{1}{t} \leq \frac{\mathrm{E}[N(t)]}{t} \leq \frac{1}{\mathrm{E}[\tilde{X}_1]} + \frac{1}{\sqrt{t}\mathrm{E}[\tilde{X}_1]} - \frac{1}{t}$$

then as $s \to \infty$ we observe that $\mathrm{E}[\tilde{X}_1] \to \mathrm{E}[X_1]$ and now letting $t \to \infty$ clearly $\mathrm{E}[N(t)]/t \to 1/\mathrm{E}[X_1]$. □

A related theorem by Blackwell gives the asymptotic rate of the expected number of renewals occurring in an interval of fixed length. We give the non-arithmetic case here (a probability distribution is arithmetic if the distribution is concentrated on a set of equally spaced points).

Theorem 1.7 (Blackwell's theorem [99])**.** *Let $X_1, X_2, \dots$ be positive* IID *random variables which are non-arithmetic with mean* $\mathrm{E}[X_1] < \infty$ *and let* $\{N(t) : t \geq 0\}$ *be the associated renewal process. Then for any* $s > 0$

$$\lim_{t\to\infty} (\mathrm{E}[N(t+s)] - \mathrm{E}[N(t)]) = \frac{s}{\mathrm{E}[X_1]} \,.$$

We note that for large t the elementary renewal theorem gives $\mathrm{E}[N(t+s)] - \mathrm{E}[N(t)] \approx (t+s)/\mathrm{E}[X_1] - t/\mathrm{E}[X_1] = s/\mathrm{E}[X_1]$. For a rigorous proof we refer the reader to [99].

Example 1.8. Let $N(t)$ be the Poisson process as in Example 1.2. One has

$$\begin{aligned}\mathrm{E}[N(t)] = \sum_{k=0}^{\infty} k \cdot \frac{(-t\log(r))^k}{k!} e^{t\log(r)} &= (-t\log(r)) e^{t\log(r)} \sum_{k=1}^{\infty} \frac{(-t\log(r))^{k-1}}{(k-1)!} \\ &= -t\log(r) e^{t\log(r)} e^{-t\log(r)} = -t\log(r) \,,\end{aligned}$$

and thus $\frac{\mathrm{E}[N(t)]}{t} = -\log(r) = \frac{1}{\mathrm{E}[X_1]}$. Notice that this is for any $t \in [0, \infty)$ as opposed to the result of Theorem 1.4. This completes the example.

The following theorem tells us about the convolution of functions with the expectation of $N(t)$ in the limit $t \to \infty$ and will be used later.

Theorem 1.9 (Key renewal theorem [99, 48]). *Let $H(t)$ be directly Riemann integrable and $H(t) = 0$ for $t < 0$, $X_1, X_2, \dots$ be positive* IID *random variables which are non-arithmetic with mean $\mathrm{E}[X_1] < \infty$ and let $\{N(t) : t \geq 0\}$ be the associated renewal process with $M(t) = \mathrm{E}[N(t)]$, then*

$$\int_0^t H(t-s)\, dM(s) \xrightarrow{t\to\infty} \frac{1}{\mathrm{E}[X_1]} \int_0^\infty H(s)\, ds\,.$$

A full proof can be found in [48].

So far we have studied the mean of $N(t)/t$, of course it would also be useful to know something about the variance. The following central limit theorem holds for renewal processes.

Theorem 1.10 (Central limit theorem for renewal processes [99]). *Let $\{N(t) : t \geq 0\}$ be a renewal process where inter-arrival times have finite standard deviation $\sigma > 0$, then*

$$\lim_{t\to\infty} \Pr\left\{ \frac{N(t) - t/\mathrm{E}[X_1]}{\sigma \mathrm{E}[X_1]^{-3/2}\sqrt{t}} < x \right\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-y^2/2}\, dy\,.$$

For a proof we refer the reader to [99]. We merely comment here that the result implies that $N(t)$ tends to a normal distribution with mean $t/\mathrm{E}[X_1]$ and variance $\sigma^2 \mathrm{E}[X_1]^{-3} t$.

The above asymptotic results are very useful when one wishes to estimate the long term consequences of failures in a system. For example, system administrators can estimate the expected cost of replacing components over a long service period t by multiplying the individual cost of a component by $N(t) \approx \frac{t}{\mathrm{E}[X_1]}$ as a result of Theorem 1.4. Similarly, a user running a job in late production (large t) for time s can estimate the expected number of failures to be $\mathrm{E}[N(t+s)] - \mathrm{E}[N(t)] \approx \frac{s}{\mathrm{E}[X_1]}$ as a result of Theorem 1.7. However, as the previous results are asymptotic ($t \to \infty$) it is not clear if they may be used as estimates for relatively small t. In Example 1.8 we saw that $\mathrm{E}[N(t)] = \frac{t}{\mathrm{E}[X_1]}$ for all $t \in [0, \infty)$ when the inter-arrival times are exponentially distributed. Thus one might expect that for distributions similar to that of the exponential distribution (for example, the Weibull distribution with shape parameter close to 1) then the asymptotic estimates may still be applied to small t with reasonable accuracy.

To study times t which are not large one has the renewal equation.

Theorem 1.11 (Renewal equation [99]). *Let $\{N(t); t \geq 0\}$ be a renewal process with mean inter-arrival times $0 < \mathrm{E}[X_1] < \infty$, then*

$$\mathrm{E}[N(t)] = F(t) + \int_0^t \mathrm{E}[N(t-s)]\, dF(s)\,, \tag{1.3}$$

where $F(t)$ is the cumulative distribution of the X_i.

Proof. From the law of total expectation $\mathrm{E}[N(t)] = \mathrm{E}_{X_1}[\mathrm{E}[N(t) \mid X_1]]$. Further, note that for $X_1 > t$ one has $\mathrm{E}[N(t) \mid X_1] = 0$ and for $X_1 \leq t$ the renewal process $N(t) - 1$ from the time X_1 is statistically identical to the renewal process $N(t)$ starting from 0, in particular $\mathrm{E}[N(t) \mid X_1 \leq t] = 1 + \mathrm{E}[N(t - X_1)]$. It follows that

$$\begin{aligned}\mathrm{E}[N(t)] = \int_0^t \mathrm{E}[N(t) \mid X_1 = s]\, dF(s) &= \int_0^t (1 + \mathrm{E}[N(t-s)])\; dF(s)\\ &= F(t) + \int_0^t \mathrm{E}[N(t-s)]\, dF(s)\,,\end{aligned}$$

as required. □

The result shows that $\mathrm{E}[N(t)]$ is the solution to a Volterra integral equation of the second kind. It is also related to a more convenient expression for $\mathrm{E}[N(t)]$. If X_1 has a probability distribution f then the renewal equation can be written as the convolution $M = F + M * f$ where $M(t) := \mathrm{E}[N(t)]$. Remember that for two independent random variables X and Y which have probability distributions f_X and f_Y respectively then the probability distribution of $X + Y$ is given by the convolution $f_{X+Y}(t) = (f_X * f_Y)(t) := \int_\infty^\infty f_X(t-s) f_Y(s) ds$. Similarly if F_X, F_Y, F_{X+Y} are the cumulative distributions of $X, Y, X+Y$ respectively then $F_{X+Y} = F_X * f_Y = f_X * F_Y$. Thus if we define $F_n := F_{n-1} * f$ with $F_1 = F$ being the cumulative density function (CDF) of the X_i then $F_{S_n} = F_{X_1 + X_2 + \cdots + X_n} = F_n$. Since $N(t) = \sum_{n=1}^\infty \chi_{[0,t]}(S_i)$ then

$$M(t) = \mathrm{E}[N(t)] = \sum_{n=1}^\infty \Pr(S_n \leq t) = \sum_{n=1}^\infty F_n(t)\,.$$

As a consistency check we note that if the probability density function (PDF) $f(t) = \frac{dF(t)}{dt}$ is well-defined then

$$\sum_{n=1}^\infty F_n = F + \sum_{n=2}^\infty F_n = F + \left(\sum_{n=1}^\infty F_n\right) * f = F + M * f = M\,.$$

The renewal equation generalises to the following result.

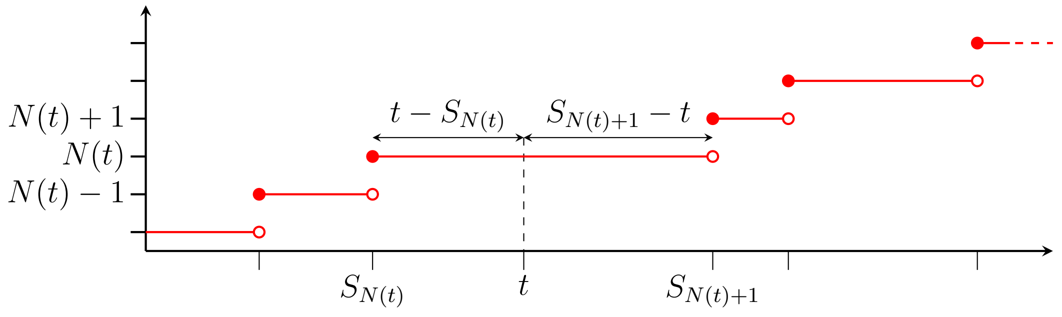


Figure 1.5: *Forward and backward recurrence times $S_{N(t)+1} - t$ and $t - S_{N(t)}$ respectively.*

Theorem 1.12 ([99]). *Given F and M defined above and*

$$h(t) = g(t) + \int_0^t h(t-s)\,dF(s)$$

for $t \geq 0$ then

$$h(t) = g(t) + \int_0^t g(t-s)\,dM(s)\,.$$

This can be proved by taking the Laplace transform, re-arranging to get an expression for h and taking the inverse Laplace transform, see for example [99].

For a user logging into a compute node at a random time the most relevant question may be: what is the expected time from now until the next failure? Equivalently, given a time t at which the user logs in, he/she would like to know $\mathrm{E}[S_{N(t)+1} - t]$. This is referred to as the forward recurrence time, or sometimes as the residual lifetime or random incidence and is depicted in Figure 1.5. The following theorem gives us an integral for calculating $\mathrm{E}[S_{N(t)+1} - t]$ for a given t as well as the asymptotic behaviour for large t.

Theorem 1.13 (Forward recurrence time [99]). *Let $X_1, X_2, \ldots$ be positive* IID *random variables with mean $0 < \mathrm{E}[X_1] < \infty$ and cumulative density function $F = F_{X_1}$. Let $S_i = \sum_{j=1}^{i} X_j$ and $N(t)$ be the associated renewal process with $M(t) := \mathrm{E}[N(t)]$. Then, for $s, t \in [0, \infty)$, the cumulative distribution of the random variable $S_{N(t)+1} - t$ is given by*

$$\Pr(S_{N(t)+1} - t \leq s) = F(t+s) - \int_0^t 1 - F(t+s-r)\,dM(r)\,. \tag{1.4}$$

Further, if the X_i are non-arithmetic then

$$\lim_{t\to\infty} \Pr(S_{N(t)+1} - t \leq s) = \frac{1}{\mathrm{E}[X_1]} \int_0^s 1 - F(r)\,dr\,. \tag{1.5}$$

Proof. One has

$$\begin{aligned}\Pr(S_{N(t)+1}-t\leq s) &= 1-\Pr(S_{N(t)+1}-t>s)\\ &= 1-\int_0^\infty \Pr(S_{N(t)+1}-t>s\mid X_1=r)\,dF(r)\,.\end{aligned}$$

If $r>t+s$ then $X_1>t$ and so $N(t)=0$. Therefore $S_{N(t)+1}=S_1=X_1=r>t+s$ and so $\Pr(S_{N(t)+1}-t>s)=1$. Similarly if $t<r\leq t+s$ then again $N(t)=0$ and therefore $S_{N(t)+1}-t=X_1-t\leq s$ and so $\Pr(S_{N(t)+1}-t>s)=0$. Lastly if $r\leq t$ then the process is restarted from $t=r$ and we have $\Pr(S_{N(t)+1}-t>s\mid X_1=r)=\Pr(S_{N(t-r)+1}-(t-r)>s)$ (i.e. shifting so that X_1 is the origin gives an equivalent renewal process). Putting the pieces together one obtains

$$\begin{aligned}\Pr(S_{N(t)+1}-t\leq s) &= 1-\int_0^t \Pr(S_{N(t-r)+1}-(t-r)>s)\,dF(r)-\int_{t+s}^\infty 1\,dF(r)\\ &= F(t+s)-\int_0^t 1-\Pr(S_{N(t-r)+1}-(t-r)\leq s)\,dF(r)\\ &= F(t+s)-F(t)+\int_0^t \Pr(S_{N(t-r)+1}-(t-r)\leq s)\,dF(r)\,.\end{aligned}$$

We can apply Theorem 1.12 to this last line to obtain

$$\Pr(S_{N(t)+1}-t\leq s)=F(t+s)-F(t)+\int_0^t F(t+s-r)-F(t-r)\,dM(r)\,. \tag{1.6}$$

Now notice that $\int_0^t F(t-r)\,dM(r)=\int_0^t M(t-r)\,dF(r)=M(t)-F(t)$ by Theorem 1.11. Substituting this into (1.6) leads to (1.4). To obtain (1.5) we apply Theorem 1.9 to (1.4) to obtain

$$\begin{aligned}F(t+s)-\int_0^t 1-F(t+s-r)\,dM(r) \xrightarrow[t\to\infty]{} &1-\frac{1}{\mathrm{E}[X_1]}\int_0^\infty 1-F(r+s)\,dr\\ =&1-\frac{1}{\mathrm{E}[X_1]}\int_s^\infty 1-F(r)\,dr\\ =&1-\left(1-\frac{1}{\mathrm{E}[X_1]}\int_0^s 1-F(r)\,dr\right)\\ =&\frac{1}{\mathrm{E}[X_1]}\int_0^s 1-F(r)\,dr\,,\end{aligned}$$

as required. □

It follows that the PDF of the asymptotic forward recurrence time is given by $\frac{1-F(s)}{\mathrm{E}[X_1]}$ and hence the expectation is

$$\lim_{t\to\infty}\mathrm{E}[S_{N(t)+1}-t]=\frac{1}{\mathrm{E}[X_1]}\int_0^\infty s(1-F(s))\,ds\,.$$

If the X_i are exponentially distributed with mean μ then one obtains

$$\begin{aligned}\lim_{t\to\infty} \mathrm{E}[S_{N(t)+1} - t] &= \mu^{-1} \int_0^\infty s\left(1 - (1 - e^{-s/\mu})\right)\, ds \\ &= \left[-se^{-s/\mu}\right]_{s=0}^{\infty} + \int_0^\infty e^{-s/\mu}\, ds = \mu\,.\end{aligned}$$

This is consistent with the memory-less property previously described. If the X_i are Weibull distributed with scale λ and shape κ then

$$\lim_{t\to\infty} \mathrm{E}[S_{N(t)+1} - t] = \frac{1}{\lambda\Gamma(1+\frac{1}{\kappa})} \int_0^\infty s\left(1 - (1 - e^{-(s/\mu)^\kappa})\right)\, ds\,,$$

and making the substitution $s \mapsto \lambda z^{1/\kappa}$ results in

$$\lim_{t\to\infty} \mathrm{E}[S_{N(t)+1} - t] = \frac{1}{\lambda\Gamma(1+\frac{1}{\kappa})} \int_0^\infty \lambda z^{1/\kappa} e^{-z} \frac{\lambda}{\kappa} z^{1/\kappa - 1}\, dz\,, = \frac{\lambda\Gamma(\frac{2}{\kappa})}{\kappa\Gamma(1+\frac{1}{\kappa})}\,.$$

If the user is able to access system logs and determine the time of last failure, that is $S_{N(t)}$, and knows something about the distribution of time between failures (i.e. the X_i and hence $X_{N(t)+1}$) then one could attempt to compute the forward recurrence time via

$$\mathrm{E}[S_{N(t)+1} - t] = \mathrm{E}[S_{N(t)} + X_{N(t)+1} - t] = \mathrm{E}[X_{N(t)+1}] - \mathrm{E}[t - S_{N(t)}]\,.$$

The trick here however is that typically $\mathrm{E}[X_{N(t)+1}] \neq \mathrm{E}[X_1]$. This is known as the inspection paradox. For example, it can be shown (similar to the proof of Theorem 1.13) that

$$\lim_{t\to\infty} \Pr(t - S_{N(t)} \leq s) = \frac{1}{\mathrm{E}[X_1]} \int_0^s 1 - F(r)\, dr\,,$$

and as a consequence

$$\mathrm{E}[X_{N(t)+1}] = \mathrm{E}[t - S_{N(t)}] + \mathrm{E}[S_{N(t)+1} - t] = \frac{2}{\mathrm{E}[X_1]} \int_0^\infty s(1 - F(r))\, ds\,.$$

For exponentially distributed X_i this evaluates to $2\mathrm{E}[X_1]$ and thus $\mathrm{E}[X_{N(t)+1}]$ is twice the expected interval length for a Poisson process. The simplest explanation of this phenomenon is that one is more likely to intercept a long interval than a short one. This result has important consequences for estimates based on observations at random times. For example, estimating the mean time between failures by taking the average over intervals which were observed at randomly chosen times will result in an overestimate.

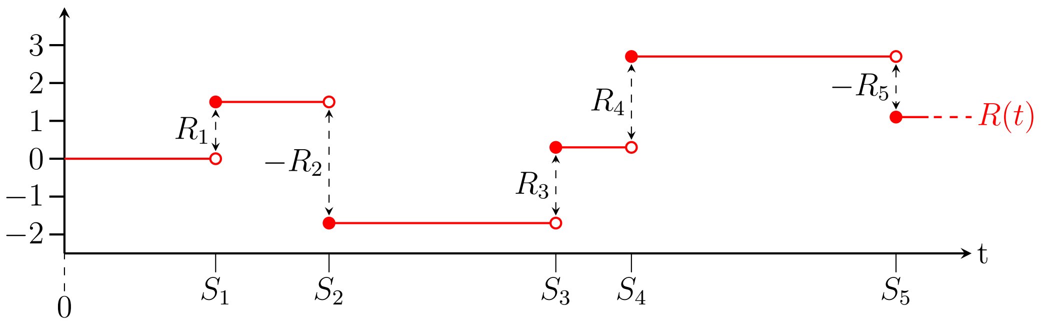


Figure 1.6: *The renewal reward model with* $R(t) = \sum_{i=1}^{\infty} R_i \chi_{[0,t]}(S_i)$.

A useful extension of the renewal process is the renewal reward process. The renewal process $N(t)$ is obtained by incrementing by 1 whenever a renewal occurs, i.e. at each S_i. Suppose instead we have a process $R(t)$ which changes according to a random variable R_i at each S_i. More formally we let R_i be IID random variables and define

$$R(t) := \sum_{i=1}^{N(t)} R_i = \sum_{i=1}^{\infty} R_i \chi_{[0,t]}(S_i) \tag{1.7}$$

with $R(t) := 0$ for $N(t) = 0$. An example of $R(t)$ is depicted in Figure 1.6. Note that the R_i need not be independent of X_i, in fact many interesting examples arise when R_i is a function of X_i. In the context of faults a renewal reward process may be serve as a model for silent errors. Whilst the X_i determine the time between errors the magnitude of the error can be modelled by the R_i. $R(t)$ then gives the sum of the errors which have occurred up to some time t thus measuring the cumulative effect. Many of the results for ordinary renewal processes can be extended to renewal-reward processes. We state a few here without proof.

Theorem 1.14 (Elementary renewal reward theorem [76, 99])**.** *Let $X_1, X_2, \ldots$ be a sequence of positive* IID *random variables with finite expectation and $N(t)$ be the associated renewal process. Further, let $R_1, R_2, \ldots$ be a second sequence of* IID *random variables with which we define the renewal reward-process $R(t) = \sum_{i=1}^{N(t)} R_i$. Then with probability* 1

$$\lim_{t\to\infty} \frac{\mathrm{E}[R(t)]}{t} = \frac{\mathrm{E}[R_1]}{\mathrm{E}[X_1]}\,.$$

Similarly we have a result analogous to Blackwell's theorem.

Theorem 1.15 ([95])**.** *Let $X_1, X_2, \ldots$ be positive* IID *random variables which are non-arithmetic with finite expectation and $N(t)$ be the associated renewal process.*

Further let $R_1, R_2, \ldots$ be IID *random variables associated with the renewal-reward process $R(t) = \sum_{i=1}^{N(t)} R_i$. Then for any $0 < s < \infty$*

$$\lim_{t\to\infty} (\mathrm{E}[R(t+s)] - \mathrm{E}[R(t)]) = \frac{s\mathrm{E}[R_1]}{\mathrm{E}[X_1]} .$$

There is also a central limit theorem for renewal-reward processes.

Theorem 1.16 ([126]). *Let $X_1, X_2, \ldots$ be a sequence of positive* IID *random variables with finite expectation and $N(t)$ be the associated renewal process. Further, let $R_1, R_2, \ldots$ be a second sequence of non-negative* IID *random variables associated with the renewal reward-process $R(t) = \sum_{i=1}^{N(t)} R_i$. If $\mathrm{E}[X_1^2], \mathrm{E}[R_1^2] < \infty$ then*

$$\lim_{t\to\infty} \Pr\left(\frac{R(t) - t\mathrm{E}[R_1]/\mathrm{E}[X_1]}{\mathrm{E}[(R_1 - X_1\mathrm{E}[R_1]/\mathrm{E}[X_1])^2]\sqrt{t/\mathrm{E}[X_1]}} \le x\right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}y^2}\, dy .$$

There are many other extensions of renewal processes worth mentioning. The first is the delayed or modified renewal process where the first arrival time has different distribution then the rest. That is $X_2, X_3, \ldots$ are IID with finite expectation but X_1 may have different distribution (but finite expectation). An example of this is where one starts observing a process with IID inter-arrival times at some random time. The first observed arrival is given by the forward recurrence time whilst subsequent arrivals are identically distributed. Most of the results for ordinary renewal processes are easily adapted to delayed renewal processes as the first arrival does not affect the asymptotic behaviour. Alternating renewal processes occur when one has two IID sequences $X_1, X_2, \ldots$ and $Y_1, Y_2, \ldots$ such that the inter-arrival time X_i is followed by Y_i which is then followed by X_{i+1} and so on. An example of this would be where the X_i are times between component failures and the Y_i are the repair times. If $N(t)$ is incremented at the end of each $X_i + Y_i$ cycle then the theory for ordinary renewal processes is again easily adapted. An example relating to faults which combine both alternating and renewal reward processes is in the estimation of cumulative downtime of a system. Given IID uptimes X_i and IID downtimes Y_i then we can consider the rewards $R_i = Y_i$ occurring at the end of each $X_i + Y_i$ cycle. $R(t)/t$ then measures the proportion of time the system was down up to time t and by applying the previous results it is straightforward to show that $\frac{\mathrm{E}[R(t)]}{t} \xrightarrow[t\to\infty]{} \frac{\mathrm{E}[Y_i]}{\mathrm{E}[X_i]+\mathrm{E}[Y_i]}$.

1.3 Models of many processor systems

We apply the models used for a single processor to the modelling of many processors, e.g. as in a HPC system.

1.3.1 Bernoulli processes

Consider again the simple models of Section 1.2.1 where the event of having a single processor either available or failed within a fixed interval of time is modelled as a Bernoulli trial $B_i \sim B(1,p)$. Now suppose we request m such identical processors to run independently in parallel for the same fixed interval of time. Let $B_{1,i}, B_{2,i}, \dots, B_{m,i}$ be the random variables for the state of each of the m processors in the ith interval. We define $D_i := \sum_{k=1}^{m} B_{k,i}$ which is equal to the number of processors which fail in the ith interval. Clearly $D_i \sim B(m,p)$ and the probability of k processors fail in the ith interval is given by

$$\Pr(D_i = k) = \binom{m}{k} p^k (1-p)^{m-k} .$$

The analysis of D_i is similar to that of the B_i in Section 1.2.1. Via the binomial formula it is straightforward to show that the expected number of failures throughout the ith interval is $\mathrm{E}[D_i] = mp$. It follows that the expected number of processors alive throughout the ith interval is $m(1-p)$. Similarly it is straightforward to show the variance is $\mathrm{Var}(D_i) = mp(1-p)$. As in the case of a single process model we may ask what is the time to first failure. This is given by $S_1 := \min \left\{ j : \sum_{i=1}^{j} D_i \geq 1 \right\}$. We note that $\Pr(D_i = 0) = (1-p)^m$ from which it follows that $\Pr(D_i \geq 1) = 1 - (1-p)^m$, therefore

$$\begin{aligned}
\mathrm{E}[S_1] &= \sum_{k=1}^{\infty} k(1-p)^{m(k-1)} \left(1 - (1-p)^m\right) \\
&= \frac{1-(1-p)^m}{(1-p)^m} \sum_{k=1}^{\infty} \sum_{j=1}^{k} (1-p)^{mk} \\
&= \frac{1-(1-p)^m}{(1-p)^m} \sum_{j=1}^{\infty} \sum_{k=j}^{\infty} (1-p)^{mk} = \sum_{j=1}^{\infty} \frac{(1-p)^{mj}}{(1-p)^m} = \frac{1}{1-(1-p)^m} .
\end{aligned}$$

As m increases the denominator approaches 1 from below and hence $E[S_1]$ gets smaller. As in the single processor model, the fact that the failure rate is constant means that the MTBF is equal to the time to first failure.

For a typical checkpoint restart approach to recovery, the failure of a single process triggers the restart of the entire system. This means that all processors are to be considered down in an interval when at least one processor fails in that interval. Thus, the expected availability of a system in these circumstances over n intervals is $n(1-p)^m$. In contrast, if the system is able to replace a failed processor without restarting all other processors then one may consider the availability to be the proportion of processors which are up over the n intervals. As the expected availability in one interval is $m(1-p)$ it follows that the availability over the n intervals is $nm(1-p)$ which is significantly larger than the availability if all processors must be restarted whenever a failure occurs, particularly for large m.

Whilst this may seem like a rather crude model it can be effective in circumstances where one repeatedly runs a computation on several processes that takes roughly the same amount of time for each iteration. One can estimate the value of p by keeping track of how often processor failures occur. For example, in weather forecasting the same computation is run several times each day, every day of the year, but with different initial conditions. It is reasonable to assume that the run times do not vary significantly for each computation and therefore this model may give a reasonable estimate once the p has been estimated.

1.3.2 Superposition of renewal processes

Suppose a high performance computer consists of m (fixed integer) processors operating in parallel (independently). We assume that each of these processors and their replacements have identical distributions of failure times. Let $N_1(t), \dots, N_m(t)$ be the renewal processes associated with the life cycle of each of the m processors. As each is processor is identical and independent each of the $N_i(t)$ can be assumed to be independent and identically distributed.

The total number of failures occurring across all m processors is given by the superposition process

$$Z(t) = \sum_{i=1}^{m} N_i(t)\,.$$

An example is depicted in Figure 1.7. In general $Z(t)$ is not a renewal process as it is typically not the case that inter-arrival times for $Z(t)$ are independent and identically distributed. However, given the identical and independent nature of the $N_i(t)$ one has $\mathrm{E}[Z(t)] = m\mathrm{E}[N_1(t)]$ and thus results relating to the expectation of ordinary renewal processes can be trivially applied to $Z(t)$. For example, it follows from the elementary renewal theorem 1.4 that

$$\lim_{t\to\infty} \frac{\mathrm{E}[Z(t)]}{t} = \frac{m}{\mathrm{E}[X]}\,,$$

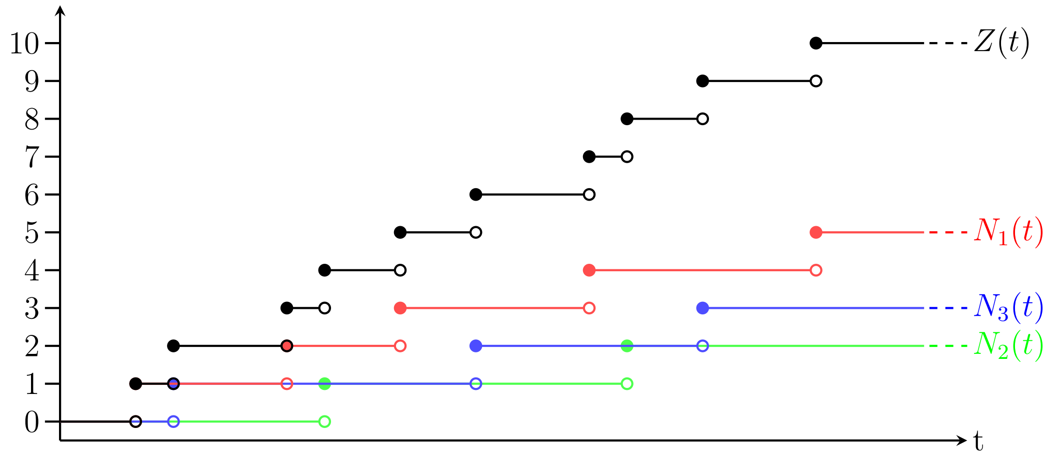


Figure 1.7: *Here we depict a stochastic process* $Z(t)$ *which is a superposition of the* 3 *renewal processes* $N_1(t)$, $N_2(t)$ *and* $N_3(t)$.

where $\mathrm{E}[X]$ is the mean inter-arrival time for all processors in the machine. Similarly Blackwell's theorem 1.7 gives

$$\lim_{t\to\infty} (\mathrm{E}[Z(t+s)] - \mathrm{E}[Z(t)]) = \frac{sm}{\mathrm{E}[X]}\,.$$

Thus we immediately see that the number of processors is proportional to the asymptotic rate at which faults occur when measure over the entire timeline and within intervals of fixed length.

Example 1.17. Suppose that the time between failures for each processor are exponentially distributed and independent with mean $\mathrm{E}[X]$. We know from Example 1.8 that the $N_i(t)$ are Poisson with $\Pr(N_i(t) = k) = \frac{(t/\mathrm{E}[X])^k}{k!}e^{-t/\mathrm{E}[X]}$. For $m = 2$ one has $Z(t) = N_1(t) + N_2(t)$ and it follows from independence that

$$\begin{aligned}
\Pr(Z(t) = k) &= \sum_{i=0}^{k} \Pr(N_1(t) = i)\Pr(N_2(t) = k - i)\\
&= \sum_{i=0}^{k} \left(\frac{(t/\mathrm{E}[X])^i}{i!}e^{-t/\mathrm{E}[X]}\right)\left(\frac{(t/\mathrm{E}[X])^{k-i}}{(k-i)!}e^{-t/\mathrm{E}[X]}\right)\\
&= (t/\mathrm{E}[X])^k e^{-2t/\mathrm{E}[X]} \sum_{i=0}^{k} \frac{1}{i!(k-i)!} = \frac{(2t/\mathrm{E}[X])^k}{k!}e^{-2t/\mathrm{E}[X]}\,,
\end{aligned}$$

where the last equality holds since $\sum_{i=0}^{k}\binom{k}{i} = 2^k$. Now consider $Z(t)$ for $m \geq 2$

and assume $\Pr(\sum_{i=1}^{m-1} N_i(t) = k) = \frac{((m-1)t/\mathrm{E}[X])^k}{k!}e^{-(m-1)t/\mathrm{E}[X]}$, then

$$\begin{aligned}
\Pr(Z(t) = k) &= \sum_{i=0}^{k} \Pr(N_m(t) = i) \Pr(N_1(t) + \cdots + N_{m-1}(t) = k - i) \\
&= \sum_{i=0}^{k} \left(\frac{(t/\mathrm{E}[X])^i}{i!} e^{-t/\mathrm{E}[X]} \right) \left(\frac{((m-1)t/\mathrm{E}[X])^{k-i}}{(k-i)!} e^{-(m-1)t/\mathrm{E}[X]} \right) \\
&= (t/\mathrm{E}[X])^k e^{-mt/\mathrm{E}[X]} \sum_{i=0}^{k} \frac{(m-1)^i}{i!(k-i)!} = \frac{(mt/\mathrm{E}[X])^k}{k!} e^{-mt/\mathrm{E}[X]} \,,
\end{aligned}$$

where the last equality holds since $\sum_{i=0}^{k}(m-1)^i\binom{k}{i} = m^k$. Thus by induction $\Pr(Z(t) = k) = \frac{(mt/\mathrm{E}[X])^k}{k!}e^{-mt/\mathrm{E}[X]}$, that is $Z(t)$ is Poisson distributed with mean m times that of $N_1(t)$. Further, this implies that $Z(t)$ is a renewal process in this particular case with inter-arrival times which are exponentially distributed with mean $\mathrm{E}[X]/m$. This concludes the example.

The central limit theorem for $N_1(t)$ says that $\mathrm{Var}(N_1(t)) \to \sigma^2 t\mathrm{E}[X]^{-3}$ as $t \to \infty$ (where $\sigma^2 = \mathrm{Var}(X)$ is the variance of the inter-arrival time for all processors in the machine). As with the expected value, this result can be extended to $Z(t)$ with

$$\Pr\left(\frac{Z(t) - mt/\mathrm{E}[X]}{\sigma\sqrt{mt}\mathrm{E}[X]^{-3/2}} \le x \right) \xrightarrow{t\to\infty} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-y^2/2}\, dy \,.$$

and thus $\mathrm{Var}(Z(t)) \to m\sigma^2 t\mathrm{E}[X]^{-3}$.

As with an ordinary renewal process, a random variable of interest may be the forward recurrence time. However as $Z(t)$ is generally not a renewal process it is not immediately clear what the forward recurrence time means. However, we may take the forward recurrence Y for $Z(t)$ to be the minimum of the forward recurrence times Y_i for $N_i(t)$, that is

$$\Pr(Y(t) \le s) = 1 - \prod_{i=1}^{m} \Pr(Y_i > s) = 1 - (1 - \Pr(Y_i \le s))^m \,.$$

For large t one may apply Theorem 1.13 to the Y_i to obtain

$$\begin{aligned}
\Pr(Y_i \le s) &\xrightarrow{t\to\infty} 1 - \left(1 - \frac{1}{\mathrm{E}[X]} \int_0^s 1 - F_X(r)\, dr \right)^m \\
&= 1 - \mathrm{E}[X]^{-m} \left(\int_s^\infty 1 - F_X(r)\, dr \right)^m \,.
\end{aligned}$$

An important question is again how large t should be for the asymptotic results to be a suitable approximation to reality. For large m one would hope that even for relatively small t that the behaviour of the m renewal processes 'averages out' such that the convergence toward asymptotic behaviour is accelerated. Thus for large HPC systems the asymptotic results may be quite accurate even for relatively small t. Whilst $Z(t)$ may not be a renewal equation in general we may consider approximating it by a renewal equation. One approach for doing this is to fit $Z(t)$ with a renewal process such that the first few moments are the same, see for example [125]. As we already know the limiting behaviour of the mean and variance of $Z(t)$ a rough approximation may already be obtained using these.

Consider processors where the time between failures is Weibull distributed. Schroeder and Gibson [116] and others have shown that the Weibull distribution is typically the best fit of time between failures from real fault data. The Weibull distribution with shape parameter κ and scale λ has the probability distribution function

$$\frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^{\kappa-1} e^{-(x/\lambda)^{\kappa}} \quad \text{for } x \geq 0\,. \tag{1.8}$$

(and 0 for $x < 0$) which has mean $\lambda\Gamma(1+1/\kappa)$ and variance $\lambda^2(\Gamma(1+2/\kappa)-\Gamma(1+1/\kappa)^2)$. The resulting $Z(t)$ asymptotically has mean $\mathrm{E}[Z(t)] \to mt/(\lambda\Gamma(1+1/\kappa))$ and variance

$$\mathrm{Var}(Z(t)) \to \frac{mt(\Gamma(1+2/\kappa)-\Gamma(1+1/\kappa)^2)}{\lambda\Gamma(1+1/\kappa)^3}\,.$$

Supposing we were to approximate $Z(t)$ with a renewal process whose inter-arrival times are also Weibull distributed with parameters κ', λ' we observe that $\kappa' = \kappa$ and $\lambda' = \lambda/m$ will give the same expectation and variance in the limit $t \to \infty$. This is consistent with the analysis of fault data by Schroeder and Gibson [116] which found that late in production the best fit of failures in a single node to be Weibull with shape 0.7 and the best fit for the entire system was also Weibull but having shape 0.78 which is close to that of a single node. Whilst the asymptotic behaviour of this approximating renewal process is the same in the first two moments, determining the how close it is to $Z(t)$ for small t requires further investigation. This is emphasised by Schroeder and Gibson fit of faults early in production which is very different from the behaviour late in production.

1.3.3 Summary

The fault models discussed here are relatively simple in nature and could be extended in many ways. For example, in practice it is unlikely that all processors operate completely independently with identical distributions for failure times. An example of dependence is that typically a fatal hardware error for a processor will affect many/all processors on the same socket and/or node. However, the models considered may be analogously applied to a socket and/or node handle this type of dependence. In practice, the fault rate may also be sensitive to the workload and therefore if workload is not evenly distributed then the fault rates are likely to differ slightly. For example Schroeder and Gibson's study [116] shows less faults occurred on weekends (when workloads are lower). Unfortunately it is not clear from their study exactly how the distribution of failure times vary with respect to workloads. It is clear that more studies need to be done into the correlation between workload and fault rate. Small variations in operating conditions could also effect the fault rate, for example the operating temperature of a node may vary slightly depending on its position in the machine room and the workload of neighbouring processors.

In this thesis we assume that during a computation workloads are relatively uniform across the processors/nodes in use and that they therefore have very similar distributions for failure times during the computation. Further, for computations on a large machine, the law of large numbers means that small perturbations in fault rates should average out and thus models based on the typical fault rate should closely approximate reality. We model the occurrence of faults on each socket/node as an ordinary renewal process with a fault on the socket/node resulting in the death of all physical processors and software processes on it. For simplicity it is assumed that t is large enough such that the asymptotic results of the renewal theory are a close approximation to reality. Future work may involve investigation into smaller t using the more general results. When a computation begins the distribution to the first failure on each node is given by the forward recurrence time. This would suggest that a delayed (or stationary) renewal process may be an appropriate model as subsequent failures will have the usual distribution. However, we argue that in practice the physical node/processors that fail will not be replaced upon failure with a new processor but rather the replacement will consist of a different node/processors in the machine which is available. Thus this replacement also has failure distribution given by the forward recurrence time. Therefore we often use the ordinary renewal process model but with inter-arrival times which are given by the forward recurrence times of another ordinary renewal process whose inter-arrival times model the time between failures.

1.4 Application of fault models

Here we show how the fault models developed can be applied to different problems affecting high performance computing. In Section 1.4.1 we look at the effect of faults on computations involving sums and averages in a parallel environment using the Bernoulli process model of faults. The results here are standard calculations of expectations and variances. In Section 1.4.2 we review the classical checkpoint restart problem of determining the times at which one should perform checkpoints to maximise utilisation. This utilises the renewal process model of failures. Whilst the result itself is relatively well-known the calculation presented here is perhaps a little more precise and detailed than that found in most references. Lastly, in Section 1.4.3 we look at the problem of fault simulation using the renewal process model of faults. To test new algorithms developed in this thesis we need to make sure they perform well when faults occur. As our computations will be done on a relatively small scale it is impractical to wait for faults to naturally occur. As such we will need to simulate faults, or at the very least their effects. We describe our approach to fault simulation for the computations performed in this thesis. It is shown that for shape parameters $\kappa \in (0,1]$ one can replace Weibull distributed times to failure with exponenitally distributed times to failure in simulations without giving our results an advantage. I have not come across the two results from this section in the literature although it is possible they are known within the probability modelling community.

1.4.1 Computation of sums and averages

Consider the computation of a sum or average of real numbers computed on different processors. Let m be the number of processors and $v_1, \dots, v_m \in \mathbb{R}$ be the m numbers with each v_i corresponding to the number computed on the ith processor. Let $u = \sum_{i=1}^m v_i$ denote the sum and thus u/m denotes the average. We assume that each of the v_i are strictly positive and therefore $u > 0$.

Suppose that some of the processors may fail and that failed processors do not contribute to the sum. Let $B_1, \dots, B_m$ be Bernoulli random variables denoting the state of each processor at the moment u is computed. Let 1 denote the 'on' state and 0 denote the 'failed' state. For each i let $p_i := \Pr(B_i = 1)$ and hence $1 - p_i = \Pr(B_i = 0)$. It follows that $\mathrm{E}[B_i] = p_i$ and $\mathrm{Var}(B_i) = p_i(1 - p_i)$. The total number of processors in the 'on' state is given by the random variable $M := \sum_{i=1}^m B_i$ and thus the number of failed processors is $m - M$. The expected number of processors contributing to the sum is $\mathrm{E}[M] = \sum_{i=1}^{m} p_i$ and the expected number of failed processors is $m - \mathrm{E}[M]$. In the case that all of the p_i are equal to some $p \in (0,1)$ then $\mathrm{E}[M] = pm\,(> 0)$ and $m - \mathrm{E}[M] = m(1-p)$.

The sum of v_i on the processors in the presence of faults is given by

$$U = \sum_{i=1}^{m} B_i v_i$$

for which one has

$$\mathrm{E}[U] = \sum_{i=1}^{m} \mathrm{E}[B_i] v_i = \sum_{i=1}^{m} p_i v_i \,.$$

If all of the p_i are equal one has $\mathrm{E}[U] = p \sum_{i=1}^{m} v_i = pu$. Thus the expected value of the sum differs from the actual sum u by $(1-p)u$ and therefore the expectation of the relative error is $1-p$.

There are two ways that one may attempt to correct the sum depending on whether all surviving processors are aware of how many processors have failed. If they are aware then one may compute $\frac{m}{M}U$ (which we define as 0 if $M = 0$). If they are not aware but $\mathrm{E}[M]$ is known one may compute $\frac{m}{\mathrm{E}[M]}U$. We first look at the expectation of these two approaches.

Proposition 1.18. *Fix $m \in \mathbb{N}$. Let $(v_1, \ldots, v_m) \in \mathbb{R}$ and $B_1, \ldots, B_m \sim B(1,p)$ be independent Bernoulli random variables with $p \in (0,1)$. Let $U = \sum_{i=1}^{m} B_i v_i$ and $M = \sum_{i=1}^{m} B_i$, then*

$$\begin{aligned} \mathrm{E}\left[\frac{mU}{\mathrm{E}[M]}\right] &= u \\ \mathrm{E}\left[\frac{mU}{M}\right] &= u\left(1 - (1-p)^m\right) \,, \end{aligned}$$

where $u = \sum_{i=1}^{m} v_i$.

Proof. For $\frac{mU}{\mathrm{E}[M]}$ we have

$$\mathrm{E}\left[\frac{m}{\mathrm{E}[M]}U\right] = \frac{m}{\mathrm{E}[M]}\mathrm{E}[U] = m\frac{\sum_{i=1}^{m} pv_i}{\sum_{i=1}^{m} p} = m\frac{pu}{pm} = u \,.$$

For $\frac{mU}{M}$ via the law of total expectation

$$\begin{aligned}
\mathrm{E}\left[m\frac{U}{M}\right] &= m\sum_{k=0}^{m}\mathrm{E}\left[\frac{U}{M}\middle|M=k\right]\Pr(M=k) \\
&= m\sum_{k=1}^{m}\mathrm{E}\left[\frac{\sum_{i=1}^{m}B_iv_i}{M}\middle|M=k\right]\Pr(M=k) \\
&= \sum_{k=1}^{m}\frac{m\Pr(M=k)}{k}\mathrm{E}\left[\sum_{i=1}^{m}B_iv_i\middle|M=k\right] \\
&= \sum_{k=1}^{m}\frac{m\Pr(M=k)}{k}\sum_{i=1}^{m}\mathrm{E}\left[B_i|M=k\right]v_i\,.
\end{aligned}$$

Now as the B_i are 1 with equal probability then given $M=k$ the probability any one B_i is 1 is $\frac{k}{m}$, that is $\Pr(B_i=1|M=k)=\frac{k}{m}$, thus $\mathrm{E}[B_i|M=k]=\frac{k}{m}$. (Alternatively, note that $\Pr(B_i=1,\,M=k)=p\binom{m-1}{k-1}p^{k-1}(1-p)^{m-k}$ and thus $\frac{\Pr(B_i=1,\,M=k)}{\Pr(M=k)}=\frac{m}{k}$). Substituting this into the previous equality one obtains

$$\mathrm{E}\left[m\frac{U}{M}\right]=\sum_{k=1}^{m}\Pr(M=k)u=u(1-\Pr(M=0))=u\left(1-(1-p)^m\right)\,,$$

as required. □

It may at first seem that $\frac{mU}{\mathrm{E}[M]}$ is the better option as the expectation of $\frac{mU}{M}$ differs from the true sum. Note, however, that the factor $(1-(1-p)^m)$ relates to the fact that the result is 0 if every processor fails and that this cannot be corrected in any way. In such circumstances one would typically start the computation again until a non-zero result is obtained. Thus one might instead consider $\mathrm{E}\left[m\frac{U}{M}\middle|M>0\right]$ which is indeed equal to u.

The formula $\frac{mU}{\mathrm{E}[M]}$ and $\frac{mU}{M}$ for a corrected sum in the event of faults may also be applied to estimate the average in the event of faults. Notice that in the case of computing an average as $\frac{U}{m}$ one has $\mathrm{E}\left[\frac{U}{m}\right]=\frac{\mathrm{E}[U]}{m}=\frac{pu}{m}$ (when all the p_i are equal to p). However, if one replaces U with the sum $\frac{m}{\mathrm{E}[M]}U$, one instead computes the average as $\frac{mU}{\mathrm{E}[M]}\frac{1}{m}=\frac{U}{\mathrm{E}[M]}$ which has mean $\mathrm{E}\left[\frac{U}{\mathrm{E}[M]}\right]=\frac{\mathrm{E}[U]}{\mathrm{E}[M]}=\frac{pu}{pm}=\frac{u}{m}$ which is the same as the average in the absence of faults. Similarly, if one uses the corrected sum $\frac{m}{M}U$ then one has the average $\frac{U}{M}$ which for which $\mathrm{E}\left[\frac{U}{M}\right]=\frac{(1-(1-p)^m)u}{m}$. Again, if we restrict ourselves to the case $M>0$ one has $\mathrm{E}\left[\frac{U}{M}\middle|M>0\right]=\frac{u}{m}$.

From the expectation alone it is not clear which of $\frac{m}{\mathrm{E}[M]}U$ or $\frac{m}{M}U$ is the better computation for a sum on a faulty machine. To get a better understanding of how far results are spread from the mean we will look at the variance of each random variable.

Proposition 1.19. *Fix $m \in \mathbb{N}$. Let $(v_1, \dots, v_m) \in \mathbb{R}$ and $B_1, \dots, B_m \sim B(1,p)$ be independent Bernoulli random variables with $p \in (0,1)$. Let $U = \sum_{i=1}^m B_i v_i$ and $M = \sum_{i=1}^m B_i$, then*

$$\begin{aligned}\operatorname{Var}\left(\frac{mU}{\mathrm{E}[M]}\right) &= \frac{1-p}{p}\sum_{i=1}^m v_i^2 \\ \operatorname{Var}\left(\frac{mU}{M}\right) &\le m\left(1-(1-p)^m\right)\left(\sum_{i=1}^m v_i^2\right) - \left(1-(1-p)^m\right)^2 u^2\,,\end{aligned}$$

where $u = \sum_{i=1}^m v_i$.

Proof. First note that

$$\operatorname{Var}(U) = \operatorname{Var}\left(\sum_{i=1}^m B_i v_i\right) = \sum_{i=1}^m v_i^2 \operatorname{Var}(B_i) = p(1-p)\sum_{i=1}^m v_i^2\,.$$

It follows that

$$\begin{aligned}\operatorname{Var}\left(\frac{m}{\mathrm{E}[M]}U\right) = \frac{m^2}{\mathrm{E}[M]^2}\operatorname{Var}(U) &= \frac{m^2}{\mathrm{E}[M]^2}p(1-p)\sum_{i=1}^m v_i^2 \\ &= \frac{1-p}{p}\sum_{i=1}^m v_i^2\,.\end{aligned}$$

For the variance of $\frac{mU}{M}$ we have

$$\begin{aligned}\operatorname{Var}\left(\frac{mU}{M}\right) &= m^2\mathrm{E}\left[\left(\frac{U}{M}\right)^2\right] - \mathrm{E}\left[\frac{mU}{M}\right]^2 \\ &= m^2\mathrm{E}\left[\left(\frac{U}{M}\right)^2\right] - \left(1-(1-p)^m\right)^2 u^2\,.\end{aligned}$$

For the $\mathrm{E}\left[\left(\frac{U}{M}\right)^2\right]$ term we have via the law of total expectation

$$\begin{aligned}\mathrm{E}\left[\left(\frac{U}{M}\right)^2\right] &= \sum_{k=0}^m \mathrm{E}\left[\left(\frac{U}{M}\right)^2 \middle| M=k\right]\Pr(M=k) \\ &= \sum_{k=1}^m \frac{1}{k^2}\mathrm{E}\left[U^2 \middle| M=k\right]\Pr(M=k) \\ &= \sum_{k=1}^m \frac{1}{k^2}\mathrm{E}\left[\left(\sum_{i=1}^m B_i v_i\right)^2 \middle| M=k\right]\Pr(M=k)\,.\end{aligned}$$

Now expanding the sum within the expectation one has

$$\mathrm{E}\left[\left(\sum_{i=1}^{m} B_i v_i\right)^2 \middle| M=k\right] = \left(\sum_{i=1}^{m} v_i^2 \mathrm{E}[B_i^2|M=k]\right) + \left(2\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} v_i v_j \mathrm{E}[B_i B_j|M=k]\right) .$$

Note that $\mathrm{E}[B_i^2|M=k] = \mathrm{E}[B_i|M=k] = \frac{k}{m}$ and since $B_iB_j = 1$ iff $B_i = B_j = 1$ one has $\Pr(B_iB_j = 1,\, M = k) = p^2\binom{m-2}{k-2}p^{k-2}(1-p)^{m-k}$ and thus $\mathrm{E}[B_iB_j|M = k] = \frac{k(k-1)}{m(m-1)}$. It follows that

$$\mathrm{E}\left[\left(\sum_{i=1}^{m} B_i v_i\right)^2 \middle| M=k\right] = \left(\frac{k}{m}\sum_{i=1}^{m} v_i^2\right) + \left(2\frac{k(k-1)}{m(m-1)}\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} v_i v_j\right) .$$

Notice that as $2v_iv_j \leq v_i^2 + v_j^2$ one has

$$\begin{aligned}\mathrm{E}\left[\left(\sum_{i=1}^{m} B_i v_i\right)^2 \middle| M=k\right] &\leq \left(\frac{k}{m}\sum_{i=1}^{m} v_i^2\right) + \left(\frac{k(k-1)}{m(m-1)}(m-1)\sum_{i=1}^{m} v_i^2\right) \\ &= \frac{k^2}{m}\sum_{i=1}^{m} v_i^2 ,\end{aligned}$$

(with equality if all of the v_i are equal). Therefore

$$\begin{aligned}\mathrm{E}\left[\left(\frac{U}{M}\right)^2\right] &\leq \frac{1}{m}\left(\sum_{i=1}^{m} v_i^2\right)\sum_{k=1}^{m}\Pr(M=k) \\ &= \frac{1}{m}\left(\sum_{i=1}^{m} v_i^2\right)(1-(1-p)^m) .\end{aligned}$$

Substituting this into the variance one has

$$\mathrm{Var}\left(\frac{mU}{M}\right) \leq m\left(1-(1-p)^m\right)\left(\sum_{i=1}^{m} v_i^2\right) - \left(1-(1-p)^m\right)^2 u^2 ,$$

as required. □

Notice that if all of the v_i are equal then $u^2 = m\sum_{i=1}^{m} v_i^2$ such that one has $\mathrm{Var}\left(\frac{mU}{M}\right) = m(1-p)^m\left(1-(1-p)^m\right)\sum_{i=1}^{m} v_i^2$ which is small for large m as a result of the $(1-p)^m$ term. One might expect the variance in this case to be zero, but the contribution here is from the case $M = 0$. One could instead consider

$$\begin{aligned}\operatorname{Var}\left(\frac{mU}{M}\middle|M>0\right) &= m^2\mathrm{E}\left[\left(\frac{U}{M}\right)^2\middle|M>0\right] - \mathrm{E}\left[\frac{mU}{M}\middle|M>0\right]^2 \\ &= m^2\mathrm{E}\left[\left(\frac{U}{M}\middle|M>0\right)^2\right] - u^2\,,\end{aligned}$$

for which $\mathrm{E}\left[\left(\frac{U}{M}|M>0\right)^2\right] \leq \frac{1}{m}\sum_{i=1}^m v_i^2$ follows from the proof of the Lemma. When all of the v_i are equal then $\operatorname{Var}\left(\frac{mU}{M}|M>0\right) = 0$. For v_i which are close together we therefore expect the variance to be small and, in particular, significantly smaller than $\operatorname{Var}(U)$ and $\operatorname{Var}\left(\frac{mU}{\mathrm{E}[M]}\right)$.

This rather simple model already gives us a nice analysis of different ways to compute the sum or average of values on a machine which may experience faults. $\frac{mU}{M}$ is typically the better estimate of the sum with $\frac{U}{M}$ being the corresponding estimate of the average. This model could easily be extended to B_i which are not IID although the calculation of mean and variance becomes increasingly complex and may have to be estimated computationally via Monte Carlo methods. One could also consider the case that the v_i themselves are random variables, e.g. they may be samples of a stochastic process or stochastic differential equation.

1.4.2 Optimal checkpoint restart algorithms

In Section 1.1.1 the checkpoint restart approach to fault tolerance was discussed. Checkpoint-restart involves periodically saving the state of the computation such that when a failure has occurred the computation can be started from the last saved state rather than from the beginning, see Figure 1.2. A classical problem in the development and analysis of checkpoint restart algorithms is the determination of the optimal checkpoint interval, that is the frequency at which checkpoints should be saved to maximise utilisation.

As in Figure 1.2 we denote s to be the initial start up time and e to be the restart time after a failure. For simplicity we will assume that $e = s$. We use c to denote the time it takes to save a checkpoint and it is assumed this is constant (i.e. does not change during the computation). The computation time before each checkpoint is denoted by r. It is r that we want to optimise with respect to the utilisation.

Let the number of faults which have occurred on a processor up to time t be modelled by a renewal process $N(t)$ with inter-arrival times given by $X_1, X_2, \ldots$

which are IID. If we start computing at some time t then the interval up until the next fault occurs, that is $[t, S_{N(t)+1})$ with $S_n := \sum_{i=1}^{n} X_i$, consists of

- an initial startup (or restart) of length s
- a number of full compute-checkpoint cycles (each of length $r+c$)
- a last compute compute-checkpoint cycle in which a failure occurs.

An exception to this is if a fault occurs during the initial startup interval in which case we must restart without having computed anything. The number of complete compute-checkpoint cycles is given by $\left\lfloor \frac{\max\{X_i - s, 0\}}{c+r} \right\rfloor$. It follows that the utilised time in the ith interval is $r \left\lfloor \frac{\max\{X_i - s, 0\}}{c+r} \right\rfloor$. Rather than maximising the utilisation we can consider the equivalent problem of minimising the waste time where the waste in each interval is given by

$$R_i = X_i - r \left\lfloor \frac{\max\{X_i - s, 0\}}{c+r} \right\rfloor .$$

Notice that as the X_i are IID the R_i are also IID so we may therefore consider a renewal reward process $R(t) = \sum_{i=1}^{N(t)} R_i$ which adds up the waste time from all intervals X_1 up to $X_{N(t)} - 1$. Note that it does not take into account any waste accumulating in the current interval $X_{N(t)}$. The elementary renewal reward theorem 1.14 tells us

$$\lim_{t \to \infty} \frac{\mathrm{E}[R(t)]}{t} \to \frac{\mathrm{E}[R_1]}{\mathrm{E}[X_1]} .$$

That is, the average rate at which the expected waste time accumulates is equal (asymptotically) to the expected rate in the first interval. As a consequence, we need only consider minimising $\mathrm{E}[R_1]$ to maximise the expected utilisation.

Proposition 1.20. *Let $s, c, r > 0$, X_1 be an exponentially distributed random variable with mean $\lambda = \mathrm{E}[X_1] > 0$, and $R_1 = X_1 - r \left\lfloor \frac{\max\{X_1 - s, 0\}}{c+r} \right\rfloor$, then*

$$\mathrm{E}[R_1] = \lambda - \frac{r e^{-s/\lambda}}{e^{(c+r)/\lambda} - 1} .$$

Proof. Via the law of total expectation

$$\mathrm{E}[R_1] = \mathrm{E}[R_1 | X_1 \le s] \Pr(X_1 \le s) + \mathrm{E}[R_1 | X_1 > s] \Pr(X_1 > s) .$$

It is straightforward to show that

$$\mathrm{E}[R_1 | X_1 \le s] \Pr(X_1 \le s) = \int_0^s \frac{x}{\lambda} e^{-x/\lambda} \, ds = \lambda - (\lambda + s) e^{-s/\lambda} .$$

For the second term we have

$$\begin{aligned}
\mathrm{E}[R_1|X_1>s] &= \mathrm{E}\left[X_1 - r\left\lfloor\frac{X_1-s}{c+r}\right\rfloor\middle|X_1>s\right] \\
&= \mathrm{E}\left[X_1+s-r\left\lfloor\frac{X_1}{c+r}\right\rfloor\middle|X_1>0\right] \\
&= \mathrm{E}\left[\frac{cX_1}{c+r}+s+r\left(\frac{X_1}{c+r}-\left\lfloor\frac{X_1}{c+r}\right\rfloor\right)\right] \\
&= \frac{c\mathrm{E}[X_1]}{c+r}+s+r\mathrm{E}\left[\frac{X_1}{c+r}-\left\lfloor\frac{X_1}{c+r}\right\rfloor\right].
\end{aligned}$$

Using the fact that X_1 is exponentially distributed one has

$$\begin{aligned}
\mathrm{E}\left[\frac{X_1}{c+r}-\left\lfloor\frac{X_1}{c+r}\right\rfloor\right] &= \int_0^\infty\left(\frac{x}{c+r}-\left\lfloor\frac{x}{c+r}\right\rfloor\right)\frac{1}{\lambda}e^{-x/\lambda}\,dx \\
&= \int_0^\infty (x-\lfloor x\rfloor)\frac{c+r}{\lambda}e^{-x(c+r)/\lambda}\,dx\,.
\end{aligned}$$

Now for each $k=0,1,2,\dots$ and $x\in[k,k+1)$ we have $\lfloor x\rfloor = k$ and so

$$\begin{aligned}
\mathrm{E}\left[\frac{X_1}{c+r}-\left\lfloor\frac{X_1}{c+r}\right\rfloor\right] &= \sum_{k=0}^\infty\left(\int_0^1 x\frac{c+r}{\lambda}e^{-(x+k)(c+r)/\lambda}\,dx\right) \\
&= \sum_{k=0}^\infty\left(e^{-k(c+r)/\lambda}\int_0^1 x\frac{c+r}{\lambda}e^{-x(c+r)/\lambda}\,dx\right) \\
&= \frac{\lambda}{c+r}\left(\int_0^{(c+r)/\lambda} xe^{-x}\,dx\right)\sum_{k=0}^\infty e^{-k(c+r)/\lambda}\,.
\end{aligned}$$

Now as $\sum_{k=0}^\infty e^{-k(c+r)/\lambda} = (1-e^{-(c+r)/\lambda})^{-1}$ and

$$\begin{aligned}
\int_0^{(c+r)/\lambda} xe^{-x}\,dx &= \left[-xe^{-x}\right]_{x=0}^{(c+r)/\lambda}+\int_0^{(c+r)/\lambda}e^{-x}\,dx \\
&= -\frac{c+r}{\lambda}e^{-(c+r)/\lambda}+\left[-e^{-x}\right]_{x=0}^{(c+r)/\lambda} = 1-\left(1+\frac{c+r}{\lambda}\right)e^{-(c+r)/\lambda}\,,
\end{aligned}$$

one has

$$\begin{aligned}
\mathrm{E}\left[\frac{X_1}{c+r}-\left\lfloor\frac{X_1}{c+r}\right\rfloor\right] &= \frac{\lambda}{c+r}\left(1-\left(1+\frac{c+r}{\lambda}\right)e^{-(c+r)/\lambda}\right)\frac{1}{1-e^{-(c+r)/\lambda}} \\
&= \frac{\lambda}{c+r}-\frac{1}{e^{(c+r)/\lambda}-1}\,.
\end{aligned}$$

Therefore

$$\mathrm{E}[R_1|X_1>s] = \frac{c\lambda}{c+r}+s+r\mathrm{E}\left[\frac{X_1}{c+r}-\left\lfloor\frac{X_1}{c+r}\right\rfloor\right] = \lambda+s-\frac{r}{e^{(c+r)/\lambda}-1}\,.$$

Substituting this into the expression for $\mathrm{E}[R_1]$ along with $\Pr(X_1>s) = e^{-s/\lambda}$ gives the desired result. □

Notice that $\mathrm{E}[R_1|X_1 > s]$ is the only term which depends on r. Thus to minimise $\mathrm{E}[R_1]$ with respect to r we need only minimise $\mathrm{E}[R_1|X_1 > s]$ with respect to r. Further, the term $\mathrm{E}\left[\frac{X_1}{c+r} - \left\lfloor\frac{X_1}{c+r}\right\rfloor\right]$ tells us how far into the last compute-checkpoint cycle we are expected to get before a failure occurs. If the fault rate is constant one might conjecture that this term is equal to $\frac{1}{2}$. In fact this is note quite true, although close if $\mathrm{E}[X_1] \ll c + r$. Using Laurent series expansion

$$\frac{1}{e^x - 1} = \frac{1}{x} - \frac{1}{2} + \frac{x}{12} - \frac{x^3}{720} + \mathcal{O}(x^5)$$

we have

$$\mathrm{E}\left[\frac{X_1}{c+r} - \left\lfloor\frac{X_1}{c+r}\right\rfloor\right] = \frac{1}{2} - \frac{c+r}{12\lambda} + \frac{(c+r)^3}{720\lambda^3} + \mathcal{O}((c+r)^5\lambda^{-5}),$$

which is approximately $1/2$ for $(c+r) \ll \lambda$. Thus

$$\mathrm{E}[R_1|X_1 > s] = \frac{c\mathrm{E}[X_1]}{c+r} + s + r\mathrm{E}\left[\frac{X_1}{c+r} - \left\lfloor\frac{X_1}{c+r}\right\rfloor\right] \approx \frac{c\mathrm{E}[X_1]}{c+r} + s + \frac{r}{2}.$$

Minimising the right hand size with respect to r we note that

$$\frac{\partial}{\partial r}\left(\frac{c\mathrm{E}[X_1]}{c+r} + s + \frac{r}{2}\right) = -\frac{c\mathrm{E}[X_1]}{(c+r)^2} + \frac{1}{2}$$

and therefore the minimum is achieved for

$$r = -c + \sqrt{2c\mathrm{E}[X_1]}.$$

As $r + c \propto \sqrt{\mathrm{E}[X_1]}$ we indeed have $c + r \ll \mathrm{E}[X_1]$ for large $\mathrm{E}[X_1]$. Thus for large $\mathrm{E}[X_1]$ the optimal compute-checkpoint interval is approximately $r+c = \sqrt{2c\mathrm{E}[X_1]}$ which is similar to approximation of the optimal checkpoint interval derived by Young [130]. The exact solution is given by $r = \lambda(W(-e^{-1-c/\lambda}) + 1)$ where W is the Lambert W function (which satisfies $W(z)e^{W(z)} = z$).

In this calculation we made two assumptions, that checkpoints occur at regular intervals, and that the X_i are exponentially distributed. More detailed calculations of optimal checkpoint intervals in the literature deal with both non-regular checkpoint times and arbitrary failure distributions, see for example [91]. Our derivation is easily applied to arbitrary failure distributions although the term $\mathrm{E}\left[\frac{X_1}{c+r} - \left\lfloor\frac{X_1}{c+r}\right\rfloor\right]$ may need to be estimated computationally. For example, if the

X_i are Weibull distributed with scale μ and shape parameter κ one has

$$\begin{aligned}
&\int_0^\infty \left(\frac{x}{c+r} - \left\lfloor\frac{x}{c+r}\right\rfloor\right) \frac{\kappa}{\mu}\left(\frac{x}{\mu}\right)^{\kappa-1} e^{-(x/\mu)^\kappa}\,dx \\
=&\int_0^\infty (x - \lfloor x \rfloor)\,\frac{\kappa(c+r)}{\mu}\left(\frac{x(c+r)}{\mu}\right)^{\kappa-1} e^{-(x(c+r)/\mu)^\kappa}\,dx \\
=&\frac{\mu}{c+r}\Gamma\left(1+\frac{1}{\kappa}\right) - \sum_{i=1}^\infty i\int_i^{i+1} \frac{\kappa(c+r)}{\mu}\left(\frac{x(c+r)}{\mu}\right)^{\kappa-1} e^{-(x(c+r)/\mu)^\kappa}\,dx \\
=&\frac{\mu}{c+r}\Gamma\left(1+\frac{1}{\kappa}\right) - \sum_{i=1}^\infty i\left(e^{-(i(c+r)/\mu)^\kappa} - e^{-((i+1)(c+r)/\mu)^\kappa}\right) \\
=&\frac{\mathrm{E}[X_1]}{c+r} - \sum_{i=1}^\infty e^{-(i(c+r)/\mu)^\kappa}\,,
\end{aligned}$$

where $\mathrm{E}[X_1] = \mu\Gamma\left(1+\frac{1}{\kappa}\right)$. Thus in this case

$$\begin{aligned}
\mathrm{E}[R_1|X_1 > s] &= \frac{c\mathrm{E}[X_1]}{c+r} + s + r\left(\frac{\mathrm{E}[X_1]}{c+r} - \sum_{i=1}^\infty e^{-(i(c+r)/\mu)^\kappa}\right) \\
&= \mathrm{E}[X_1] + s - r\sum_{i=1}^\infty e^{-(i(c+r)/\mu)^\kappa}\,.
\end{aligned}$$

One then would need to find the minimum with respect to r for a given μ,κ and c which could be done computationally (using Newton's method for example).

1.4.3 Fault simulation

In Section 1.3.3 some assumptions that are made regarding faults and fault models in this thesis were discussed. Here we make some additional comments regarding our approach to the simulation of hard faults (i.e. where a processor or node stops working permanently). As previously mentioned, at the current time there is limited software available which notifies applications of a fault so that the application itself can decide what action to take. User Level Fault Mitigation (ULFM) is an extension of MPI which allows an application to detect the loss of an MPI processor and then shrink the global communicator or even replace the lost processor. Although ULFM is still in beta some use cases are emerging [103, 4]. Because software like ULFM is still in early development it is difficult to test software recovering from real faults. Further, limited resources mean we cannot test our algorithms at the peta- to exa- scale waiting for real faults. The difficulty in testing faults at a smaller scale is that faults are less frequent. As our main interest is algorithm development testing is essential. Thus, our approach will be to artificially raise the fault rate on smaller problems via fault simulation with small MTBF such that the rate at which faults occur on the smaller problems is similar, or even more frequent, than that for larger problems. Also note that the loss of data on one node for a problem running on $\mathcal{O}(10)$ nodes is more significant than a loss of one node for a problem running on $\mathcal{O}(10\,000)$ or more nodes. Therefore, with good results in these circumstances we can be confident that the algorithm will perform well at a larger scale.

We will make the following assumptions regarding simulated hard faults.

- affected processors fail permanently, all data on the affected processors is lost,
- a software mechanism is in place to replace the failed processors (by other processors on standby) and this is transparent to the application,
- the application is notified that a fault has occurred and which processors were affected,
- other processors are not aware of the failure until they attempt to communicate with other processors (failed or otherwise).

This is simulated by deleting all data on the MPI processors for which a simulated failure was flagged, from that point on we assume the affected MPI processors have been transparently replaced by a supporting software mechanism. Only at a communication block of code does the application become aware of the failure.

In a real computer system most permanent failures will result in the loss of an entire socket or node. With a modular system design one could argue that sockets should be able to operate independently and therefore only a socket should be typically affected by hardware failure. Thus, assuming that future hardware is implemented in such a way to reduce the impact of hard faults, we model the life cycle of each socket as a renewal process. This is simulated by having one thread on each socket sample an appropriate distribution for the time to the next failure. A flag is then raised so that at the next communication block of code the application becomes aware of the failure. All threads on that socket are notified that the data is void. The thread responsible for fault simulation on that socket then enters a new interval in the renewal process and thus samples another time to next failure. The application then enters into some recovery scheme to replace or recompute the lost data which may or may not involve processors from other sockets (which also become aware of the fault at the next communication block). From here computations continue as usual until completion or the occurrence of another fault.

For the distribution of time between failures we focus on the Weibull distribution (1.8) with shape parameter $0 < \kappa \leq 1$. The reason for this is that the two studies [116] and [91] of faults in high performance computers found the Weibull distribution to be the best fit and the former study found both individual nodes and entire systems exhibited shape parameters of approximately 0.7 and 0.8 respectively in late production. There are a few important observations to be made for the Weibull distribution with shape $0 < \kappa \leq 1$. First, for $\kappa = 1$ it reduces to the exponential distribution. Second, for $\kappa < 1$, unlike the exponential distribution which is memory-less, the failure rate (or hazard rate) decreases over time. In particular one has the following lemma.

Proposition 1.21. *If X is Weibull distributed with $\kappa \in (0,1]$ and $s,t \geq 0$ then*

$$\Pr(X \leq s+t \mid X > s) \leq \Pr(X \leq t)\,. \tag{1.9}$$

Proof. One has

$$\Pr(X \leq s+t \mid X > s) = \frac{\Pr(s < X \leq s+t)}{\Pr(X > s)}\,.$$

Noting that a Weibull random variable has cumulative distribution

$$\Pr(X \leq t) = 1 - e^{-(t/\lambda)^\kappa} \quad \text{for } t \geq 0\,, \tag{1.10}$$

(and $\Pr(X \leq t) = 0$ for $t < 0$) one has

$$\Pr(X \leq s+t \mid X > s) = \frac{e^{-(s/\lambda)^\kappa} - e^{-((s+t)/\lambda)^\kappa}}{e^{-(s/\lambda)^\kappa}} = 1 - e^{(s/\lambda)^\kappa - ((s+t)/\lambda)^\kappa}\,.$$

As $\kappa \in (0,1]$ and $s,t \geq 0$ one has $s^\kappa - (s+t)^\kappa \geq -t^\kappa$ and hence

$$\Pr(X \leq s+t \mid X > s) \leq 1 - e^{-(t/\lambda)^\kappa}\,,$$

as required. □

One can easily extend this property to the fact that for $s_2 \geq s_1 \geq 0$ one has

$$\Pr(X \leq s_2 + t \mid X > s_2) \leq \Pr(X \leq s_1 + t \mid X > s_1)\,.$$

This has important implications on the order in which we compute successive solutions on a single node. Solutions one is least concerned about not completing due to a fault should be computed first and solutions for which we would like to minimise the chance of failure should be computed last. For $\kappa > 1$ we note that these inequalities are reversed, that is failure becomes more likely as time moves forward.

Remember, the forward recurrence time is the time to next failure given a starting time t, that is $Y = S_{N(t)+1} - t$. The cumulative distribution of the forward recurrence time for an ordinary renewal process is given by (1.4). We assume that t is sufficiently large that the limiting distribution (1.5) for $t \to \infty$ is a good approximation to the forward recurrence time. If the inter-arrival times are Weibull one has the limiting distribution

$$\Pr(Y \leq s) = \frac{1}{\lambda\Gamma(1+\frac{1}{\kappa})} \int_0^s e^{-(x/\lambda)^\kappa}\, dx \quad (t \geq 0)\,. \tag{1.11}$$

Thus, given a computation starting at a random time in a processors life cycle, we can sample this distribution to obtain a time until the next failure for our simulation.

Recall that for $\kappa = 1$ one has $\Pr(Y \leq s) = \Pr(X \leq s) = 1 - e^{-s/\lambda}$. The following lemma shows that $\Pr(X \leq s)$ is an upper bound for $\Pr(Y \leq s)$ if $\kappa \in (0, 1]$.

Proposition 1.22. *Let Y be the forward recurrence time for the ordinary renewal process with inter-arrival times given by the Weibull distributed random variable X having shape parameter $\kappa \in (0, 1]$. Then*

$$\Pr(Y \leq t) \leq \Pr(X \leq t)\,, \tag{1.12}$$

for all t.

Proof. We note that via a change of variables followed by integration by parts that

$$\begin{aligned}\Gamma\left(1+\frac{1}{\kappa}\right) = \int_0^\infty y^{\frac{1}{\kappa}} e^{-y}\, dy &= \int_0^\infty \frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^\kappa e^{-(x/\lambda)^\kappa}\, dx \\ &= \int_0^\infty \left(\frac{x}{\lambda}\right)\left(\frac{\kappa}{\lambda}\left(\frac{x}{\lambda}\right)^{\kappa-1} e^{-(x/\lambda)^\kappa}\right) dx \\ &= \left[-\frac{x}{\lambda}e^{-(x/\lambda)^\kappa}\right]_0^\infty - \int_0^\infty -\frac{1}{\lambda}e^{-(x/\lambda)^\kappa}\, dx\,.\end{aligned}$$

Note that for $x = 0$ one has $\frac{x}{\lambda}e^{-(x/\lambda)^\kappa} = 0$. Further, note that $e^z \geq \frac{z^m}{m!}$ for any $m \in \mathbb{N}$ and $z \geq 0$, and thus by fixing $m > 1/\kappa$ and $z = x/\lambda$ we have

$$\lim_{x\to\infty} \frac{x/\lambda}{e^{(x/\lambda)^\kappa}} \leq \lim_{x\to\infty} \frac{zm!}{z^{m\kappa}} = m! \lim_{x\to\infty} z^{1-m\kappa} = 0\,,$$

as $1 - m\kappa < 0$. As $\lim_{x\to\infty} \frac{x/\lambda}{e^{(x/\lambda)^\kappa}}$ is also clearly bounded below by zero the limit itself must be equal to zero. Therefore one has $\left[-\frac{x}{\lambda}e^{-(x/\lambda)^\kappa}\right]_0^\infty = 0$ and thus

$$\Gamma\left(1+\frac{1}{\kappa}\right) e^{-(t/\lambda)^\kappa} = \int_0^\infty \frac{1}{\lambda} e^{-(x/\lambda)^\kappa-(t/\lambda)^\kappa} dx\,.$$

Since $0 < \kappa \leq 1$ and $t, x \geq 0$ one has $x^\kappa + t^\kappa \geq (x+t)^\kappa$ and hence

$$\begin{aligned}
\Gamma\left(1+\frac{1}{\kappa}\right) e^{-(t/\lambda)^\kappa} &\leq \int_0^\infty \frac{1}{\lambda} e^{-((x+t)/\lambda)^\kappa} dx \\
&= \int_t^\infty \frac{1}{\lambda} e^{-(x/\lambda)^\kappa} dx \\
&= \int_0^\infty \frac{1}{\lambda} e^{-(x/\lambda)^\kappa} dx - \int_0^t \frac{1}{\lambda} e^{-(x/\lambda)^\kappa} dx \\
&= \Gamma\left(1+\frac{1}{\kappa}\right) - \frac{1}{\lambda}\int_0^t e^{-(x/\lambda)^\kappa} dx\,.
\end{aligned}$$

Rearranging gives

$$\frac{1}{\lambda\Gamma(1+\frac{1}{\kappa})}\int_0^t e^{-(x/\lambda)^\kappa} dx \leq 1 - e^{-(t/\lambda)^\kappa}$$

which is the desired inequality. □

This last property is extremely useful as it says that for $0 < \kappa \leq 1$, if we perform our fault simulations sampling the Weibull distribution instead of the corresponding forward recurrence time the simulated faults will occur more frequently. Thus, if we simplify our simulation by sampling X instead of Y we will not be giving ourselves an advantage when $0 < \kappa \leq 1$ and the simulation is in fact signalling faults occurring more often than in reality. If a fault tolerant algorithm performs well under these circumstances then we can be confident that it will work well in practice.

The remaining question is how might we sample the variables X and Y in a simulation. The Weibull distributed random variable X is straightforward to sample via the inverse transform sampling method. Note that the CDF of X has the simple form $F_X(x) = 1 - e^{-(x/\lambda)^\kappa}$ which has inverse $F_X^{-1}(y) = \lambda\left(-\log(1-y)\right)^{1/\kappa}$. We may therefore compute a sample t of X by computing a sample y from $U([0,1))$ (the uniform distribution on $[0,1)$) and then setting $t = F_X^{-1}(y)$. The forward recurrence time Y is somewhat more difficult to sample. As the cumulative distribution of Y is defined by an integral it is difficult to invert. Here we will outline how samples may be drawn for $\frac{1}{2} \leq \kappa \leq 1$ using acceptance-rejection sampling. Notice that we may obtain the PDF f_Y of Y by differentiating the CDF

$$f_Y(s) = \frac{d}{ds}F_Y(s) = \frac{d}{ds}\int_0^s \frac{e^{-(x/\lambda)^\kappa}}{\lambda\Gamma(1+\frac{1}{\kappa})}\,ds = \frac{e^{-(s/\lambda)^\kappa}}{\lambda\Gamma(1+\frac{1}{\kappa})}\,.$$

Now we need to find a function $g(s)$ such that $f_Y(s) \leq g(s)$ for all $s \in [0,\infty)$ and such that samples from Z having probability distribution $g(s)/\int_0^\infty g(x)\,dx$ are easy to calculate. We consider the function

$$g(s) = \frac{1}{\lambda\Gamma(1+\frac{1}{\kappa})}\left(e^{-s/\lambda} + e^{-\sqrt{s/\lambda}}\right)\,.$$

Notice that for $0 \leq s \leq \lambda$ one has $e^{-s/\lambda} \geq e^{-(s/\lambda)^\kappa}$ and for $s \geq \lambda$ one has $e^{-\sqrt{s/\lambda}} \geq e^{-(s/\lambda)^\kappa}$. It follows that $g(s) \geq f(s)$ for all $s \in [0,\infty)$. Further, the normalising factor for g is given by

$$\begin{aligned}\int_0^\infty g(s)\,ds &= \frac{1}{\lambda\Gamma(1+\frac{1}{\kappa})}\int_0^\infty e^{-s/\lambda} + e^{-\sqrt{s/\lambda}}\,ds \\ &= \frac{1}{\lambda\Gamma(1+\frac{1}{\kappa})}\left(\lambda + \int_0^\infty e^{-\sqrt{s/\lambda}}\,ds\right)\,.\end{aligned}$$

For the remaining integral we substitute $s \mapsto z^2\lambda$ to obtain

$$\begin{aligned}\int_0^\infty e^{-\sqrt{s/\lambda}}\,ds &= \int_0^\infty e^{-z}2z\lambda\,dz \\ &= \left[-2\lambda z e^{-z}\right]_{z=0}^\infty + \int_0^\infty 2\lambda e^{-z}\,dz = 0 + \left[-2\lambda e^{-z}\right]_{z=0}^\infty = 2\lambda\,,\end{aligned}$$

and therefore $\frac{\Gamma(1+\frac{1}{\kappa})}{3}g(s)$ has the necessary properties of a probability density function on $[0,\infty)$. Further, one has

$$\frac{\Gamma(1+\frac{1}{\kappa})}{3}g(s) = \frac{1}{3}\left(\frac{1}{\lambda}e^{-s/\lambda}\right) + \frac{2}{3}\left(\frac{1}{2\lambda}e^{-\sqrt{s/\lambda}}\right)$$

and thus $\frac{\Gamma(1+\frac{1}{\kappa})}{3}g(s)$ is a mixture distribution with $\frac{1}{\lambda}e^{-s/\lambda}$ corresponding to the PDF of an exponentially distributed random variable Z_1 with mean λ and $\frac{e^{-\sqrt{s/\lambda}}}{2\lambda}$ corresponding to the PDF of a random variable Z_2 (note that Z_2 has identical distribution to Y when $\kappa = \frac{1}{2}$). One can sample Z as follows:

1. Sample u from $U([0,1))$,
2. If $u < 1/3$ then sample Z_1, otherwise sample from Z_2.

Sampling from Z_1 is straightforward using the inverse transform approach (in fact it is identical to sampling the Weibull distribution described above with $\kappa = 1$). Sampling from Z_2 can also be done with the inverse transform method but requires a root finding algorithm to find the inverse. Notice that the CDF of Z_2 is given by

$$\begin{aligned}\Pr(Z_2 \le s) &= \frac{1}{2\lambda}\int_0^s e^{-\sqrt{x/\lambda}}\,dx \\ &= \frac{1}{2\lambda}\int_0^{\sqrt{s/\lambda}} e^{-z} 2\lambda z\,dz \\ &= \left[-ze^{-z}\right]_{z=0}^{\sqrt{s/\lambda}} + \int_0^{\sqrt{s/\lambda}} e^{-z}\,dz \\ &= -\sqrt{\frac{s}{\lambda}}e^{-\sqrt{s/\lambda}} + 1 - e^{-\sqrt{s/\lambda}} = 1 - e^{-\sqrt{s/\lambda}}\left(1 + \sqrt{s/\lambda}\right).\end{aligned}$$

Thus we may sample from Z_2 as follows:

1. Sample u from $U([0,1))$,
2. Solve $1 - u = e^{-z}(1+z)$ for z, for example, using Newton's method $z_{n+1} = z_n + 1 + \frac{1}{z_n} - \frac{1-u}{z_n e^{-z_n}}$ (a good initial guess is $z_0 = 1 - \log(1-u)$),
3. Return the sample λz^2.

Now that we have the function $g(s) \ge f_Y(s)$ and a straightforward way of sampling the random variable Z which has PDF $\frac{\Gamma(1+\frac{1}{\kappa})}{3} g(s)$ we can use acception-rejection sampling to sample Y as follows:

1. Take a random sample z from Z,
2. Take a random sample u from $U((0,1))$
3. If $ug(z) \le f(z)$ then accept z as a sample of Y, otherwise return to step 1.

The performance of this approach depends upon how often samples are rejected. The unconditional acceptance probability is given by

$$\begin{aligned}\Pr\left(u < \frac{f(z)}{g(z)}\right) = \mathrm{E}\left[\frac{f(z)}{g(z)}\right] &= \int_0^\infty \frac{f(z)}{g(z)}\frac{\Gamma(1+\frac{1}{\kappa})}{3} g(z)\,dz \\ &= \frac{\Gamma(1+\frac{1}{\kappa})}{3}\int_0^\infty f(z)\,dz = \frac{\Gamma(1+\frac{1}{\kappa})}{3}.\end{aligned}$$

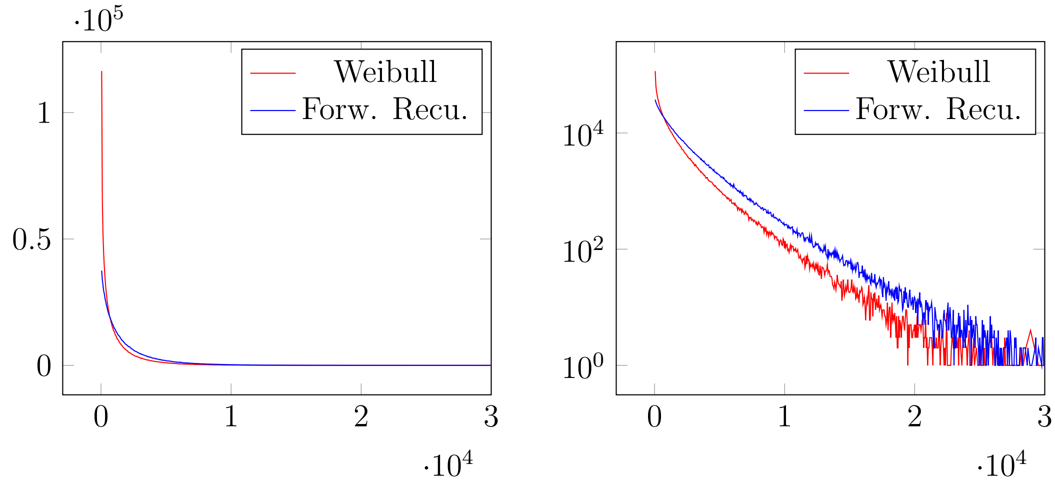


Figure 1.8: *Here we plot a histogram of 10^6 samples using 10^3 bins of the random variables X and Y corresponding to the Weibull distribution and forward recurrence times respectively with $\lambda = 1000$ and $\kappa = 0.7$. The plot on the right is of the same data but with a log scale on the y axis.*

As $\kappa \in [1/2, 1]$ one has $1 \leq \Gamma(1 + \frac{1}{\kappa}) \leq 2$ and therefore the unconditional acceptance probability is in the interval $\left[\frac{1}{3}, \frac{2}{3}\right]$. With an acceptance probability of $\frac{1}{3}$ the expected number of iterations before a sample is accepted is 3 $(\sum_{k=1}^{\infty} \frac{k}{3} \left(\frac{2}{3}\right)^{k-1} = 3)$. Suppose the computation time is proportional to the number of times the uniform distribution on the interval is sampled, then, as sampling Z requires 2 samples (one to determine which of Z_1 or Z_2 is to be sampled and the second for the actual sampling), we see that the expected number of samples of $U([0,1))$ to obtain a sample of Y is $3(2+1) = 9$.

Note that this sampling method is easily extended to κ in each sub-interval $\left[\frac{1}{n}, \frac{1}{n-1}\right]$ for integers $n \geq 2$ by considering $g(s) = \frac{1}{\lambda\Gamma(1+\frac{1}{\kappa})}\left(e^{-s/\lambda} + e^{-(s/\lambda)^{1/n}}\right)$. Here one finds that $\int_0^\infty g(s)\,ds = \frac{1+n!}{\Gamma(1+\frac{1}{\kappa})}$ which leads to the acceptance probability $\frac{\Gamma(1+\frac{1}{\kappa})}{1+n!}$ which lies in the interval $\left[\frac{(n-1)!}{1+n!}, \frac{n!}{1+n!}\right]$. However, as the study by Schroeder and Gibson [116] did not fit any shape parameters outside the interval $\kappa \in [\frac{1}{2}, 1]$ to the distribution of time between failures for computers at LANL, we restrict our simulations to shape parameters within this interval.

We conclude with an experiment comparing the distributions X and Y. In Figure 1.8 we plot the histograms obtained by taking 10^6 samples of X and Y for $\kappa = 0.7$ and $\lambda = 1000$. On the left we plot the raw histogram data and on the right we plot the log of the histogram data to make the difference between the two clearer. The random variable X has significantly more samples in the first few bins and then quickly drops below the number of samples of Y for the majority of the remaining bins. The mean and standard deviation of the samples of X are $\lambda\Gamma(1+\frac{1}{\kappa}) \approx 1268.35$ and $\lambda\sqrt{\Gamma(1+\frac{2}{\kappa}) - \Gamma(1+\frac{1}{\kappa})^2} \approx 1852.83$ respectively (compared to the true expectation and standard deviation ≈ 1265.82 and ≈ 1851.17 respectively). The mean and standard deviation of the samples of Y are 1988.65 and 2426.15 respectively (compared to the true expectation and std. dev. $\frac{\lambda\Gamma(\frac{2}{\kappa})}{\kappa\Gamma(1+\frac{1}{\kappa})} \approx 1986.51$ and $\frac{\lambda}{\kappa\Gamma(1+\frac{1}{\kappa})}\sqrt{\kappa\Gamma(1+\frac{1}{\kappa})\Gamma(\frac{3}{\kappa}) - \Gamma(\frac{2}{\kappa})^2} \approx 2420.48$ respectively). The observation that the mean of the samples of Y is larger than that of X is consistent with the result of Proposition 1.22.

Chapter 2

Sparse Grids and the Combination Technique

In order to develop fault tolerant algorithms based upon the sparse grid combination technique it is essential to give a thorough description of both sparse grids and the (sparse grid) combination technique. Many important results from the existing literature which are presented as the techniques used to derive these results will be useful in the analysis of generalisations of the combination technique and, in particular, fault-tolerant adaptations of it. We review many of the classical results from the literature in the context of a slightly different convention for combination level. The techniques used in the proofs are also of interest as these will be useful in the analysis of generalisations of the combination technique and, in particular, fault-tolerant adaptations of it. We start by introducing sparse grids in Section 2.1. In particular we will motivate their development with a discussion of high dimensional problems and the hierarchical basis representation of functions. By studying the contributions of different hierarchies in relation to the number of unknowns in each we develop sparse grids in a manner similar to the original literature on the subject. However, unlike existing literature where sparse grids are typically developed for functions which are zero on the boundary of the domain, we develop a more general case in which the boundary values are not necessarily zero. We follow this with Section 2.2 which introduces the combination technique as a way to approximate sparse grid solutions without having to work directly with a hierarchical basis. An important observation will be the combination technique's equivalence to an inclusion-exclusion principle applied to tensor products of function spaces which motivates the study of adaptive sparse grids in Section 4.2. We also point out some attractive computational features and inherent redundancies in the combination technique which make the combination technique a good starting point for developing fault-tolerant algorithms.

2.1 Sparse Grids

It is first useful to introduce some notation from the theory of partially ordered sets, or posets, which is used throughout this section and the remainder of the thesis.

Notation 2.1. We consider multi-indices $\underline{i}, \underline{j} \in \mathbb{N}^d$ with the partial ordering $\underline{i} \le \underline{j}$ iff $i_k \le j_k$ for all $k = 1, \dots, d$ (where $\underline{i} = (i_1, \dots, i_d)$). Then,

- the strict inequality $\underline{i} < \underline{j}$ is equivalent to $\underline{i} \le \underline{j}$ and $\underline{i} \ne \underline{j}$,
- the meet (or greatest lower bound) $\underline{l} = \underline{i} \wedge \underline{j}$ is the largest element satisfying $\underline{l} \le \underline{i}$ and $\underline{l} \le \underline{j}$. In $\mathbb{N}^d$ it is simply the component-wise minimum of the two indices, that is $l_k = \min\{i_k, j_k\}$ for $k = 1, \dots, d$,
- Similarly the join (or least upper bound) $\underline{i} \vee \underline{j}$ is the smallest element satisfying $\underline{l} \ge \underline{i}$ and $\underline{l} \ge \underline{j}$. In $\mathbb{N}^d$ it is also the component-wise maximum of the two indices, that is $l_k = \max\{i_k, j_k\}$ for $k = 1, \dots, d$,
- For $a \in \mathbb{N}$ we write $\underline{a}$ as shorthand for $(a, \dots, a) \in \mathbb{N}^d$, for example $\underline{0} = (0, \dots, 0)$, $\underline{1} = (1, \dots, 1)$ and $\underline{2} = (2, \dots, 2)$ is frequently used.
- $|\underline{i}|$ denotes the sum $\sum_{k=1}^{d} |i_k|$.[1]

2.1.1 Preliminaries and motivation

Sparse grids were introduced by Zenger in 1990 [131] and subsequently developed by him and Griebel [56, 62]. They were born from the realisation that for sufficiently smooth problems a full grid resolves many high frequencies which have a relatively small contribution to the solution. Zenger demonstrated that by not resolving these frequencies one could drastically reduce the cost of computing a solution whilst having a relatively small impact on the solution error. Some of the underlying ideas can be traced back to Smolyak [119] in the study of quadrature for tensor products of functions spaces.

[1]Note that as $\underline{i} \in \mathbb{N}^d$ the $|\cdot|$ is redundant. However, as we consider multi-indices with negative components later in the thesis this definition is preferred.

Bungartz also made several contributions [17, 20, 21] particularly in extending the ideas to higher order methods in the early 2000s. In 2004 Bungartz and Griebel published a survey of sparse grids [22] which is the canonical reference for the subject. More recently Garcke has published a much condensed introduction to sparse grids [50]. Of course many others also made significant contributions, particularly looking at extensions and generalisations of the initial concept, but we defer discussion of these contributions to Chapter 4. In this section we focus on the development of classical sparse grids.

We start by introducing the function space H^2_{mix}.

Definition 2.2. Given $\Omega \subset \mathbb{R}^d$, a real valued function $u \in L^2(\Omega)$ and $s \in \mathbb{N}$, then $u \in H^s_{\text{mix}}(\Omega)$ if for each $\underline{0} \leq \underline{i} \leq \underline{s}$ the weak derivative $D^{\underline{i}}u$ exists and has finite L^2 norm. This function space may be equipped with the norm

$$\|u\|^2_{H^s_{\text{mix}}(\Omega)} := \sum_{\underline{0} \leq \underline{i} \leq \underline{s}} \left\| \frac{\partial^{|\underline{i}|}}{\partial \boldsymbol{x}^{\underline{i}}} u \right\|^2_{L_2(\Omega)} = \sum_{\underline{0} \leq \underline{i} \leq \underline{s}} \left\| D^{\underline{i}} u \right\|^2_{L_2(\Omega)} .$$

Additionally, for $\underline{i} \in \mathbb{N}^d$ we define the semi-norm

$$|u|_{H^{\underline{i}}_{\text{mix}}(\Omega)} = \left\| \frac{\partial^{|\underline{i}|}}{\partial \boldsymbol{x}^{\underline{i}}} u \right\|_{L_2(\Omega)} = \left\| D^{\underline{i}} u \right\|_{L_2(\Omega)} .$$

Where the domain is clear we drop the Ω. The subset of H^s_{mix} consisting of functions which are 0 on the boundary is denoted by $H^s_{0,\text{mix}}$. Of particular interest is the function space H^2_{mix} for which we note that $H^2_{\text{mix}}(\Omega) \subset H^2(\Omega) \subset L^2(\Omega)$.

Consider the approximation of functions defined on closed intervals of the real line. Without loss of generality we will always assume our interval to be $[0, 1]$ as any other interval can be easily transformed to this via a dilation and translation operation. Let $u : [0, 1] \mapsto \mathbb{R}$ be a real valued function which is continuous and bounded. Note that we could just as easily consider complex valued functions but for simplicity we consider real valued functions in this thesis.

In order to approximate a continuous function u on a computer one typically begins by discretising the domain. In particular, it is common to break the interval up into evenly spaced/sized elements. For example, in finite volume methods an interval is typically broken into sub-intervals of equal length called cells in which one considers the function average within each cell, and in finite difference methods one typically considers the value of the function at evenly spaced points along the entire interval (and interpolates between these points). In this thesis we are interested in finite difference methods for hyperbolic PDEs and therefore consider evenly spaced points within the interval but we note that the development is much the same for finite volume methods. Furthermore, it is also commonplace to consider nested coarsening and refinements for which the distance between points (or cell widths) for different discretisations differs by a power of 2. In light of this we introduce some notation.

Notation 2.3. Given the domain $\Omega = [0, 1]$ and $i \in \mathbb{N}$ we denote Ω_i to be the discretisation of Ω into $2^i + 1$ evenly spaced points (including endpoints). That is

$$\Omega_i := \{x_{i,j} \mid j = 0, 1, \dots, 2^i\}$$

where $x_{i,j} := j2^{-i}$. Ω_i is referred to as a level i discretisation of the interval.

A function $u : \Omega \mapsto \mathbb{R}$ may be approximated by a piecewise linear function $u_i : \Omega \mapsto \mathbb{R}$ whose function values at $x \in \Omega_i$ approximate those of u. In this section we specifically consider u_i satisfying $u_i(x) = u(x)$ for all $x \in \Omega_i$. There are many ways in which one might construct such functions. We are particularly interested in those which are linear[2] on the sub-intervals $[x_{i,j}, x_{i,j+1}]$ for $j = 0, \dots, 2^i - 1$. These may be described as a linear combination of nodal basis functions.

Definition 2.4. The nodal basis function $\phi_{i,j}$ is defined as

$$\phi_{i,j}(x) := \begin{cases} 1 - 2^i|x - x_{i,j}| & x \in [x_{i,j} - 2^{-i}, x_{i,j} + 2^{-i}] \cap [0,1] \\ 0 & \text{otherwise} \end{cases} .$$

We see that for each $x_{i,j}$ the corresponding nodal basis function $\phi_{i,j}$ satisfies $\phi_{i,j}(x_{i,j}) = 1$ and $\phi_{i,j}(x_{i,k}) = 0$ for all $k \neq j$. Further, each nodal basis function is clearly linear on the sub-intervals $[x_{i,j}, x_{i,j+1}]$ for all $j = 0, \dots, 2^i - 1$. As such we may define piecewise linear interpolants as follows.

Definition 2.5. The piecewise linear interpolant of samples of $u : \Omega \mapsto \mathbb{R}$ on the set of points $x \in \Omega_i$ is denoted by

$$\mathcal{I}_i u := \sum_{j=0}^{2^i} u(x_{i,j})\phi_{i,j}(x) . \tag{2.1}$$

We refer to $\mathcal{I}_i u$ as the piecewise linear interpolant of u on Ω_i.

The function space of all such approximations is given by the span of the nodal basis functions.

[2]By 'linear' here we really mean 'affine'. As it is generally well understood what this means we use the term linear for consistency with the existing literature.

Definition 2.6. The space of piecewise linear functions which is linear on the sub-intervals $[x_{i,j}, x_{i,j+1}]$ for all $j = 0, \dots, 2^i - 1$ is defined as

$$V_i := \operatorname{span}\{\phi_{i,j} : j = 0, \dots, 2^i\}\,.$$

The nodal basis functions have the following properties (adapted from [22, 50] to include the boundary nodes).

Lemma 2.7. *Let $i \in \mathbb{N}$, if $i > 0$ and $j \in \{0, \dots, 2^i\}$, then*

$$\|\phi_{i,j}\|_1 = 2^{-i}\,, \quad \|\phi_{i,j}\|_2 = \left(\frac{2}{3}\right)^{1/2} 2^{-i/2}\,, \quad \textit{and} \quad \|\phi_{i,j}\|_\infty = 1\,.$$

Additionally, if $i = 0$ and $j \in \{0, 1\}$ then

$$\|\phi_{i,j}\|_1 = \frac{1}{2}\,, \quad \|\phi_{i,j}\|_2 = \left(\frac{1}{3}\right)^{1/2}\,, \quad \textit{and} \quad \|\phi_{i,j}\|_\infty = 1\,.$$

Proof. Straightforward evaluation of integrals for the first two and the last follows from the fact that the maximum of each $\phi_{i,j}$ is 1. □

Of course, nodal basis functions are well known and are the cornerstone of approximation theory being commonly used for numerical approximation of functions. For example, nodal basis functions are widely used in the finite element method to solve the Galerkin formulation of a variety of partial differential equation. Whilst for sufficiently large i the u_i described by (2.1) are typically a reasonable approximation of $u \in L_2(\Omega)$ it is worth pointing out that it is typically not the best approximation of u in V_i. For example, the best approximation in V_i for $u \in L_2(\Omega)$ is given by

$$\min_{v \in V_i} \|u - v\|_2\,,$$

and the study of such projections from L_2 to V_i is fundamental to finite element methods. The representation of elements of V_i via the nodal basis is attractive computationally as it typically leads to sparse linear systems of equations that may be solved quickly and efficiently. For example, given the nodal basis functions $\phi_{i,j}$ and $\phi_{i,k}$ for $j, k \in \{0, \dots, 2^i\}$ the supports overlap iff $|j - k| \leq 1$. This leads to tri-diagonal stiffness matrices for a large class of problems in one spatial dimension.

We now consider an alternative description of our approximations u_i.

Definition 2.8. Given the index sets

$$B_l = \left\{ \begin{array}{ll} 1,3,5,\dots,2^l-3,2^l-1 & l>0 \\ 0,1 & l=0 \end{array} \right\} .$$

then the level i hierarchical basis functions are the $\phi_{i,j}$ with $j \in B_i$. Further, the space of functions spanned by these basis functions

$$W_i := \operatorname{span}\{\phi_{i,j} : j \in B_i\}$$

is referred to as the space of level i hierarchical surpluses (or ith hierarchical space for short).

We claim now that $u_i = \mathcal{I}_i u$ can be written as the sum

$$u_i = \sum_{l=0}^{i} \sum_{j\in B_l} c_{l,j}\phi_{l,j}$$

for some appropriate choice of $c_{l,j} \in \mathbb{R}$.

We first note that each of the $\phi_{l,j}$ with $l < i$ has wider support than the typical nodal basis functions. However, like the nodal basis functions for each $x_{i,k}$ ($\in \Omega_i$) there is a $\phi_{l,j}$ ($0 \le l \le i$ and $j \in B_l$) such that $\phi_{l,j}(x_{i,k}) = 1$, in particular $\phi_{l,j}(x_{i,2^{i-l}j}) = 1$. Note that given the definition of $x_{i,j}$ one can write $x_{i,2^{i-l}j} = x_{l,j}$ in which case one has $\phi_{l,j}(x_{l,j}) = 1$. We also notice that $\phi_{l',j'}(x_{l,j}) = 0$ for all $l' > l$ and $j' \in B_{l'}$. This suggests a so-called 'bottom-up' approach. We note that $\phi_{0,0}(a) = \phi_{0,1}(b) = 1$ and $\phi_{0,0}(b) = \phi_{0,1}(a) = 0$ whilst all of the other $\phi_{l,j}$ ($l > 0$) are 0 at both a and b. It follows that $c_{0,0} = u(x_{0,0}) = u(x_{i,0})$ and $c_{0,1} = u(x_{0,1}) = u(x_{i,2^i})$. For $l = 1$ we have only the one hierarchical basis function $\phi_{1,1}$. Since $c_{0,0}\phi_{0,0} + c_{0,1}\phi_{0,1}$ gives us a linear interpolant between a and b then we must have

$$u(x_{1,1}) = \frac{u(x_{0,0}) + u(x_{0,1})}{2} + c_{1,1}\phi_{1,1}(x_{1,1})$$

from which it follows that

$$c_{1,1} = u(x_{1,1}) - \frac{u(x_{0,0}) + u(x_{0,1})}{2} = u(x_{i,2^{i-1}}) - \frac{u(x_{i,0}) + u(x_{i,2^i})}{2} .$$

With induction on l it is similarly shown that

$$c_{l,j} = u(x_{l,j}) - \frac{u(x_{l,j-1}) + u(x_{l,j+1})}{2} = u(x_{i,2^{i-l}j}) - \frac{u(x_{i,2^{i-l}(j-1)}) + u(x_{i,2^{i-l}(j+1)})}{2} ,$$

for $j \in B_l$ and $l = 0, \dots, i$. This is typically expressed as an operator for which we write

$$c_{l,j} = [\ -\tfrac{1}{2} \quad 1 \quad -\tfrac{1}{2}\ ]_{x_{l,j},l} u := u(x_{l,j}) - 0.5(u(x_{l,j} - 2^{-l}) + u(x_{l,j} + 2^{-l}))$$

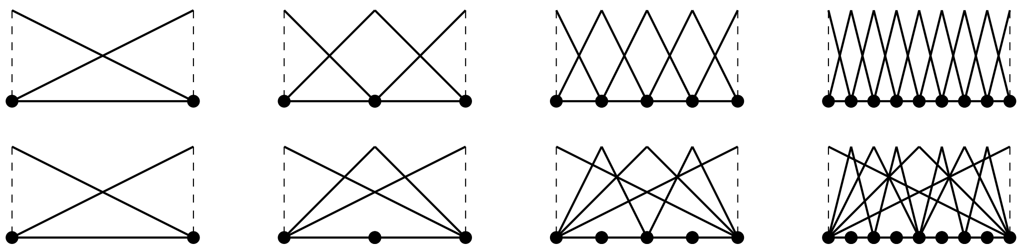


Figure 2.1: *Here we depict the linear nodal and hierarchical basis functions in one dimension for increasing levels of discretisation. The top row shows the nodal basis functions from level 0 on the left up to level 3 on the right. The bottom row shows the hierarchical basis functions for the same levels.*

By construction the resulting function is clearly equal to u_i. Further, we can write

$$V_i = \bigoplus_{l=0}^{i} W_i = \operatorname{span}\{\phi_{l,j} : l = 0, \dots, i \text{ and } j \in B_l\},$$

from which it follows that

$$V_i = V_{i-1} \oplus W_i .$$

Figure 2.1 shows the hierarchical basis functions for level 0,1,2 and 3 discretisation of an interval. The large support of many of the hierarchical basis functions is evident in the figure. As a result one typically obtains relatively dense linear systems to solve compared to the sparse systems obtained from the nodal basis function representation.

The following result provides an integral formula for the hierarchical coefficients.

Lemma 2.9 ([22, 50])**.** *Let $u \in H^2(\Omega)$ and $c_{l,j}$ be hierarchical coefficients such that $u_i = \mathcal{I}_i u = \sum_{l=0}^{i} \sum_{j \in B_l} c_{l,j}\phi_{l,j}$, then for $l > 0$ one has*

$$c_{l,j} = -\frac{2^{-l}}{2} \int_\Omega \phi_{l,j} \frac{\partial^2 u}{\partial x^2} dx$$

Proof. Using integration by parts one obtains

$$\begin{aligned}\int_\Omega \phi_{l,j} \frac{\partial^2 u}{\partial x^2}\, dx &= \left[\phi_{l,j}\frac{\partial u}{\partial x}\right]_{x_{l,j-1}}^{x_{l,j+1}} - \int_{x_{l,j-1}}^{x_{l,j+1}} \frac{\partial \phi_{l,j}}{\partial x}\frac{\partial u}{\partial x}\, dx \\ &= 0 - \frac{1}{2^{-l}} \int_{x_{l,j-1}}^{x_{l,j}} \frac{\partial u}{\partial x}\, dx + \frac{1}{2^{-l}} \int_{x_{l,j}}^{x_{l,j+1}} \frac{\partial u}{\partial x}\, dx \\ &= \frac{1}{2^{-l}} \left(-(u(x_{l,j}) - u(x_{l,j-1})) + (u(x_{l,j+1}) - u(x_{l,j}))\right) \\ &= -\frac{2}{2^{-l}} \left(u(x_{l,j}) - \frac{1}{2}(u(x_{l,j-1}) + (u(x_{l,j+1}))\right) = -\frac{2}{2^{-l}} c_{l,j} .\end{aligned}$$

Multiplying both sides by $\frac{-2^{-l}}{2}$ gives the required result. Observe that the above calculation holds as $H^2(\Omega) \subset C^1(\Omega)$ (in particular the Sobolev embedding theorem [1] gives $H^2(\Omega) \subset C^{1,\frac{1}{2}}(\Omega)$ with $C^{k,\alpha}(\Omega)$ being the Hölder space for which the kth partial derivative exists and is Hölder continuous with exponent α). $\square$

Lemma 2.9 shows that as l increases the hierarchical coefficients decrease like $\mathcal{O}(2^{-l})$, in fact as the support of the integral term is also $\mathcal{O}(2^{-l})$ the hierarchical coefficients decay even faster. This decay in hierarchical coefficients is similar to the decay of Fourier coefficients in smooth functions. This suggests that just as the tail of a Fourier series may be truncated to approximate functions with a small amount of data a similar approach may be taken with hierarchical coefficients. Note that since the support of the corresponding $\phi_{l,j}$ also decreases we see that the contribution to u_i from each W_l decreases exponentially as l increases (see Lemma 2.7). At the same time however the number of $\phi_{l,j}$ which span a given W_l increases, in particular it is equal to the size of B_l for which $|B_l| = 2^{l-1}$ for $l > 0$ and $|B_0| = 2$. It follows that as we refine an approximation to u the cost increases exponentially whilst the benefit decreases exponentially. It is with this observation that we build up sparse grids for functions in two or more dimensions.

Consider $\Omega \subset \mathbb{R}^d$ for a fixed integer d. In particular we consider domains which are topologically equivalent to tensor products of closed intervals (possibly with identification of opposite edges/faces), that is after an appropriate transformation we may write

$$\Omega = \mathfrak{I}_1 \times \cdots \times \mathfrak{I}_d$$

for some closed intervals $\mathfrak{I}_1, \ldots, \mathfrak{I}_d \subset \mathbb{R}$. Without loss of generality it is enough to consider $\Omega = [0,1]^d$ (e.g. by applying an affine transformation to $\mathfrak{I}_1 \times \cdots \times \mathfrak{I}_d$). The fact that $\Omega = [0,1]^d$ is simply a tensor product of unit intervals $\mathfrak{I} = [0,1]$ means that much of what we developed in one dimension may be extended to Ω via tensor products.

Again we consider functions $u : \Omega \mapsto \mathbb{R}$ which are continuous and bounded. To approximate the function u we start by discretising our domain. Let $\underline{i} \in \mathbb{N}^d$, then

$$\Omega_{\underline{i}} := \Omega_{i_1} \times \cdots \times \Omega_{i_d}$$

where each Ω_{i_k} is a discretisation of the unit interval $\mathfrak{I} = [0,1]$ as in Notation 2.3. In particular

$$\Omega_{\underline{i}} = \{\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d : x_k = j_k \cdot 2^{-i_k} \text{ for } j_k = 0, \ldots, 2^{i_k} \text{ and } k = 1, \ldots, d\}\,.$$

Similarly we use the notation $x_{\underline{i},\underline{j}}$ for elements of $\Omega_{\underline{i}}$ where

$$x_{\underline{i},\underline{j}} = (x_{i_1,j_1}, \dots, x_{i_d,j_d}) = (j_1 2^{-i_1}, \dots, j_d 2^{-i_d})$$

where $j_k \in \{0, \dots, 2^{i_k}\}$ for each $k = 1, \dots, d$.

Consider piecewise (multi-)linear approximations of $u : \Omega \to \mathbb{R}$ which are (multi-)linear on each $[x_{i_1,j_1}, x_{i_1,j_1+1}] \times \cdots \times [x_{i_d,j_d}, x_{i_d,j_d+1}]$. Such functions are again typically represented as a sum of (d dimensional) nodal basis functions.

Definition 2.10. The d dimensional nodal basis functions are defined by

$$\phi_{\underline{i},\underline{j}}(\boldsymbol{x}) := \prod_{k=1}^{d} \phi_{i_k,j_k}(x_k)\,.$$

Noting that each $\phi_{\underline{i},\underline{j}}$ is 1 at $x_{\underline{i},\underline{j}}$ and 0 for all other points in $\Omega_{\underline{i}}$ we can express an approximation to u which is a (multi-)linear interpolation of samples of u on Ω_i as

$$u_{\underline{i}}(\boldsymbol{x}) = \mathcal{I}_{\underline{i}} u(\boldsymbol{x}) := \sum_{(\underline{0}\leq)\,\underline{j}\leq 2^{\underline{i}}} u(x_{\underline{i},\underline{j}})\phi_{\underline{i},\underline{j}}(\boldsymbol{x})\,, \tag{2.2}$$

where $2^{\underline{i}} := (2^{i_1}, \dots, 2^{i_d})$.

Notation 2.11. We denote the space of piecewise (multi-)linear functions given by the span of the $\phi_{\underline{i},\underline{j}}$ by

$$V_{\underline{i}} := \operatorname{span}\{\phi_{\underline{i},\underline{j}} : (\underline{0} \leq)\, \underline{j} \leq 2^{\underline{i}}\}\,.$$

Given the product nature of the nodal basis functions and the space Ω it follows that $V_{\underline{i}}$ can be expressed as a tensor product of our one dimensional V_i, that is

$$V_{\underline{i}} = V_{i_1} \otimes \cdots \otimes V_{i_d}\,.$$

As before, $\mathcal{I}_i u$ is not necessarily the best approximation of u in $V_{\underline{i}}$.

We extend Lemma 2.7 to our d-dimensional nodal basis functions (adapted from [22, 50] to include the boundary nodes).

Lemma 2.12. *Let* $\underline{i} \in \mathbb{N}^d$ *and* $(\underline{0} \leq)\, \underline{j} \leq 2^{\underline{i}}$, *then*

$$\begin{aligned}
\|\phi_{\underline{i},\underline{j}}\|_1 &= 2^{-(d-|\underline{i}|_0)} 2^{-|\underline{i}|} \\
\|\phi_{\underline{i},\underline{j}}\|_2 &= \left(\frac{1}{3}\right)^{d/2} 2^{|\underline{i}|_0/2} 2^{-|\underline{i}|/2} \\
\|\phi_{\underline{i},\underline{j}}\|_\infty &= 1\,,
\end{aligned}$$

where $|\underline{i}|_0$ *is the number of non-zero entries in* $\underline{i}$.

Proof. Using the product structure of $\phi_{\underline{i},\underline{j}}$ in Definition 2.10 each equality is obtained by taking the product of the results of Lemma 2.7. □

As in the one-dimensional case, for a given $\underline{i}$ approximations with the nodal basis functions typically lead to relatively sparse linear systems since the support of $\phi_{\underline{i},\underline{j}}$ for each $(\underline{0} \leq)\, \underline{j} \leq 2^{\underline{i}}$ overlaps with at most 3^d neighbouring basis functions (including itself). However, we again consider a hierarchical description of our function space $V_{\underline{i}}$.

Notation 2.13. Let $\underline{l} \in \mathbb{N}^d$ then we define the (multi-)index set

$$B_{\underline{l}} := B_{l_1} \times \cdots \times B_{l_d} = \left\{ \underline{j} : \begin{array}{ll} j_k = 1, 3, 5, \dots, 2^{l_k} - 1 & \text{if } l_k > 0,\ k = 1, \dots, d \\ j_k = 0, 1 & \text{if } l_k = 0,\ k = 1, \dots, d \end{array} \right\},$$

and let

$$W_{\underline{l}} = \operatorname{span}\{\phi_{\underline{l},\underline{j}} : \underline{j} \in B_{\underline{l}}\}$$

Just as the $V_{\underline{i}}$ is equal to $V_{i_1} \otimes \cdots \otimes V_{i_d}$ one similarly has

$$W_{\underline{l}} = W_{l_1} \otimes \cdots \otimes W_{l_d}\,.$$

Furthermore, it follows that

$$V_{\underline{i}} = \bigotimes_{k=1}^{d} V_{i_k} = \bigotimes_{k=1}^{d} \left(\bigoplus_{l_k=0}^{i_k} W_{l_k} \right) = \bigoplus_{\underline{0} \leq \underline{l} \leq \underline{i}} \left(\bigotimes_{k=1}^{d} W_{l_k} \right) = \bigoplus_{\underline{0} \leq \underline{l} \leq \underline{i}} W_{\underline{l}}\,.$$

Henceforth we drop the lower bound $\underline{0}$ for sums over $\underline{l} \leq \underline{i}$ as this is implicit for $\underline{l} \in \mathbb{N}^d$.

The obvious question at this point is how one calculates coefficients $c_{\underline{l},\underline{j}}$ such that

$$u_{\underline{i}} = \sum_{\underline{l} \leq \underline{i}} \left(\sum_{\underline{j} \in B_{\underline{l}}} c_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}} \right).$$

The tensor product structure is again invaluable as we can simply apply our one dimensional hierarchisation operator to each dimension. That is, for $\underline{l} \geq \underline{1}$

$$c_{\underline{l},\underline{j}} = \left(\prod_{k=1}^{d} [\ -\tfrac{1}{2}\quad 1\quad -\tfrac{1}{2}\ ]_{x_{l_k,j_k},l_k} \right) u\,.$$

At this point it is important to note that the literature generally considers the domain $[0,1]^d$ with the additional assumption that functions are zero on the boundary. This somewhat simplifies the development and also fits in with the application of elliptic PDEs with zero boundary which is typically presented in the literature. In particular this means that they only need to consider $\underline{l} \geq \underline{1}$. With our interest in hyperbolic PDEs however, we must consider non-zero boundary conditions and therefore must consider hierarchical basis functions which are non-zero on the boundary. Let

$$H_{x_{l,j},l} := \begin{cases} \begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}_{x_{l,j},l} & \text{if } l > 0 \\ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}_{x_{l,j},l} & \text{if } l = 0 \end{cases}$$

then one has for all $\underline{l} \geq \underline{0}$

$$c_{\underline{l},\underline{j}} = \left(\prod_{k=1}^{d} H_{x_{l_k,j_k},l_k} \right) u \,.$$

We now extend Lemma 2.9 to our d-dimensional hierarchical coefficient $c_{\underline{l},\underline{j}}$.

Lemma 2.14 ([22, 50])**.** *Let $\Omega = [0,1]^d$ and $u \in H^2_{mix}(\Omega)$ and $c_{l,j}$ be hierarchical coefficients such that $u_{\underline{i}} = \mathcal{I}_{\underline{i}} u = \sum_{\underline{l} \leq \underline{i}} \sum_{\underline{j} \in B_{\underline{l}}} c_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}}$, then for $\underline{l} \geq \underline{1}$ one has*

$$c_{\underline{l},\underline{j}} = (-1)^d 2^{-|\underline{l}|-d} \int_\Omega \phi_{\underline{l},\underline{j}} D^{\underline{2}} u \, d\boldsymbol{x} \,. \tag{2.3}$$

Additionally for $\underline{l} \not\geq \underline{1}$ (and $\underline{l} \geq \underline{0}$), let k be the number of non-zero members of $\underline{l}$ and $\{m_1, \dots, m_k\} \subset \{1, \dots, d\}$ be such that $l_{m_1}, \dots, l_{m_k} \neq 0$ and $\{m_{k+1}, \dots, m_d\}$ are the remaining indices, then one has

$$c_{\underline{l},\underline{j}} = (-1)^k 2^{-|\underline{l}|-k} \int_0^1 \cdots \int_0^1 \phi_{\underline{l},\underline{j}} \frac{\partial^2}{\partial x_{m_1}^2} \cdots \frac{\partial^2}{\partial x_{m_k}^2} u|_{x_{m_{k+1}}=j_{k+1},\dots,x_{m_d}=j_d} \, dx_{m_1} \cdots dx_{m_k} \,. \tag{2.4}$$

Proof. Given the product structure of both $\phi_{\underline{l},\underline{j}}$ and $c_{\underline{l},\underline{j}}$ then (2.3) follows immediately from Lemma 2.9 (observing that $H^2_{\text{mix}}([0,1]^d) = H^2([0,1]) \otimes \cdots \otimes H^2([0,1])$, see [73]). Similarly (2.4) is the same argument applied to those indices which are non-zero. □

For $\underline{l} \leq \underline{i}$ we use the notation $u^h_{\underline{l}}$ to denote the contribution to the function $u_{\underline{i}}$ from the hierarchical space $W_{\underline{l}}$, that is $u^h_{\underline{l}} \subset W_{\underline{l}}$. As such $u_{\underline{i}} = \sum_{\underline{l} \leq \underline{i}} u^h_{\underline{l}}$. The following Lemma bounds the contribution from hierarchical surpluses and is effectively a collection of two separate lemmas in [22].

Lemma 2.15 ([22, 50])**.** *Let $u \in H^2_{mix}$ and for $\underline{i} \in \mathbb{N}^d$ let $u_{\underline{i}} = \mathcal{I}_{\underline{i}}$ be piecewise linear approximations of u as in (2.2). For $\underline{l} \leq \underline{i}$ and $\underline{j} \in B_{\underline{l}}$ let $c_{\underline{l},\underline{j}}$ be such that $u_{\underline{i}} = \sum_{\underline{l} \leq \underline{i}} \left(\sum_{\underline{j} \in B_{\underline{l}}} c_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}} \right)$, then for $\underline{l} \geq \underline{1}$*

$$\left\| c_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}} \right\|_2 \leq 3^{-d} 2^{-2|\underline{l}|} \left\| D^{\underline{2}} u|_{supp(\phi_{\underline{l},\underline{j}})} \right\|_2 \,.$$

Additionally, if $u_{\underline{l}}^h \in W_{\underline{l}}$ are the hierarchical surpluses such that $u_{\underline{i}} = \sum_{\underline{l} \leq \underline{i}} u_{\underline{l}}^h$, then for $\underline{l} \geq \underline{1}$ one has

$$\left\| u_{\underline{l}}^h \right\|_2 \leq \left(\frac{1}{3}\right)^d 2^{-2|\underline{l}|} \left\| D^{\underline{2}} u \right\|_2 .$$

Proof. Combining Lemmas 2.12 and 2.14 for $\underline{l} \geq \underline{1}$ (for which $|\underline{l}|_0 = d$) one has (via Hölder's inequality)

$$\begin{aligned}
\left\| c_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}} \right\|_2^2 = \left| c_{\underline{l},\underline{j}} \right|^2 \left\| \phi_{\underline{l},\underline{j}} \right\|_2^2 &= \left(\frac{1}{6}\right)^d 2^{-3|\underline{l}|} \left| \int_\Omega \phi_{\underline{l},\underline{j}} D^{\underline{2}} u(\boldsymbol{x}) d\boldsymbol{x} \right|^2 \\
&\leq \left(\frac{1}{6}\right)^d 2^{-3|\underline{l}|} \left\| \phi_{\underline{l},\underline{j}} \right\|_2^2 \left\| D^{\underline{2}} u|_{\mathrm{supp}(\phi_{\underline{l},\underline{j}})} \right\|_2^2 \\
&= \left(\frac{1}{9}\right)^d 2^{-4|\underline{l}|} \left\| D^{\underline{2}} u|_{\mathrm{supp}(\phi_{\underline{l},\underline{j}})} \right\|_2^2 .
\end{aligned}$$

Summing over all $\underline{j} \in B_{\underline{l}}$ for a given $\underline{l} \geq \underline{1}$ we note that the supports of the $\phi_{\underline{l},\underline{j}}$ overlap only on their boundaries and the union is Ω, thus one obtains the bound

$$\left\| u_{\underline{l}}^h \right\|_2 = \left\| \sum_{\underline{j} \in B_{\underline{l}}} c_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}} \right\|_2 \leq \left(\frac{1}{3}\right)^d 2^{-2|\underline{l}|} \left\| D^{\underline{2}} u \right\|_2 ,$$

as required. □

We see that the contribution to the approximation from the hierarchical surplus $u_{\underline{l}}^h$ decays exponentially according to the 1-norm of $\underline{l}$. At the same time, the number of hierarchical basis functions making up $u_{\underline{l}}^h$ is given by $|B_{\underline{l}}| = 2^{|\underline{l}|-d}$ for $\underline{l} \geq \underline{1}$. Thus the cost increases exponentially whilst the benefit decreases exponentially. We see that the $u_{\underline{l}}^h$ with the same cost and benefit are those with equal $|\underline{l}|$. The idea of a sparse grid is that rather than consider a full grid solution of level n, that is $u_{(n,\dots,n)} = \sum_{\underline{l} \leq (n,\dots,n)} u_{\underline{l}}^j$ we instead consider $\sum_{|\underline{l}| \leq n} u_{\underline{l}}^h$. As will be quantified in in the following subsections, this significantly reduces the number of unknowns in the representation whilst only having small impact on the error.

Notation 2.16. We denote the sparse grid function space by V_n^s and it can be expressed in terms of its hierarchical decomposition into

$$V_n^s = \bigoplus_{|\underline{l}| \leq n} W_{\underline{l}} .$$

Similarly we denote $u_n^s := \sum_{|\underline{l}| \leq n} u_{\underline{l}}^h$ to be an approximation of the function u in the sparse grid space. It is typically referred to as the sparse grid solution.

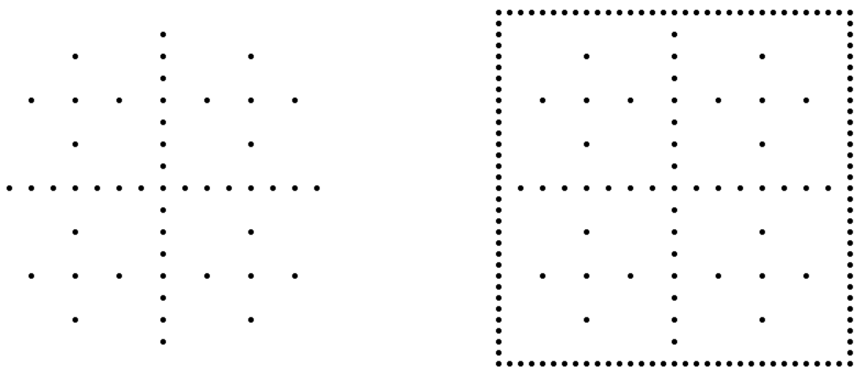


Figure 2.2: *Here we depict a two dimensional sparse grid. On the left is a sparse grid without boundary points and on the right is the same sparse grid with boundary points.*

The largest hierarchical spaces in V_n^s have $\mathcal{O}(2^n)$ unknowns and contribute $\mathcal{O}(2^{-2n})$ to the solution. Compared to a full grid where the largest hierarchical space contains approximately $\mathcal{O}(2^{dn})$ unknowns and contributes only $\mathcal{O}(2^{-2dn})$ to the solution we already have some indication that the sparse grid is more efficient. Figure 2.2 depicts the sample points in a classical sparse grid in which it is evident that the total number of points is far less than those of the smallest full grid containing the sparse grid in each case. In the next section we quantify the number of unknowns in a sparse grid. Following this we consider the error $\|u - u_n^s\|_2$ allowing us to quantify the intuition gained thus far.

2.1.2 Number of unknowns in a sparse grid

On a level n full grid it is clear that we have $(2^n + 1)^d$ if the grid includes points on the boundary and $(2^n - 1)^d$ if it does not include points on the boundary. In either case we can see that the number of unknowns is approximately 2^{dn}. We first consider a sparse grid without boundary points as in [22, 50]. In literature referring to sparse grids without boundaries it is typical to 'shift' the origin so that what they refer to as a level n sparse grid is actually level $n + d - 1$ in the notation introduced here. This shift accounts for the fact that $\underline{1}$ is the smallest hierarchical level in this case. We will not do this and thus the result here differs slightly.

Proposition 2.17 ([50]). *The number of unknowns in a level n sparse grid with-*

out boundaries is given by

$$(-1)^d + 2^{n-d+1} \sum_{k=0}^{d-1} \binom{n}{d-1-k} (-2)^k \,. \tag{2.5}$$

Proof. Given no boundary points we note that the hierarchical levels of interest are $\underline{l} \geq \underline{1}$ (point-wise) and that for such $\underline{l}$ one has

$$|B_{\underline{l}}| = 2^{l_1-1} \times \cdots \times 2^{l_d-1} = 2^{|\underline{l}|-d} \,.$$

Therefore the number of unknowns are

$$\sum_{\underline{l}\geq\underline{1},\, |\underline{l}|\leq n} |B_{\underline{l}}| = \sum_{k=d}^{n} \left(\sum_{\underline{l}\geq\underline{1},\, |\underline{l}|=k} 2^{k-d} \right) = \sum_{k=d}^{n} \binom{k-1}{d-1} 2^{k-d} = \sum_{k=0}^{n-d} \binom{k+d-1}{d-1} 2^k \,.$$

Following from our last equation we have

$$\begin{aligned}
\sum_{k=0}^{n-d} \binom{k+d-1}{d-1} 2^k &= \frac{1}{(d-1)!} \sum_{k=0}^{n-d} \left(x^{k+d-1}\right)^{(d-1)} \Big|_{x=2} \\
&= \frac{1}{(d-1)!} \left(\sum_{k=0}^{n-d} \left(x^{k+d-1}\right) \right)^{(d-1)} \Bigg|_{x=2} \\
&= \frac{1}{(d-1)!} \left(\frac{x^{d-1} - x^n}{1-x} \right)^{(d-1)} \Bigg|_{x=2} ,
\end{aligned}$$

where the notation $f(x)^{(a)}$ denotes the ath derivative of $f(x)$ with respect to x. Applying the product rule we obtain

$$\begin{aligned}
&\sum_{k=0}^{n-d} \binom{k+d-1}{d-1} 2^k \\
=&\frac{1}{(d-1)!} \sum_{k=0}^{d-1} \left(\binom{d-1}{k} \left(x^{d-1} - x^n\right)^{(k)} \left(\frac{1}{1-x} \right)^{(d-1-k)} \right) \Bigg|_{x=2} \\
=&\frac{1}{(d-1)!} \sum_{k=0}^{d-1} \binom{d-1}{k} \left(\frac{x^{d-1-k}(d-1)!}{(d-1-k)!} - \frac{x^{n-k} n!}{(n-k)!} \right) \left(\frac{(d-1-k)!}{(1-x)^{d-k}} \right) \Bigg|_{x=2} \\
=&\sum_{k=0}^{d-1} \left(\binom{d-1}{k} x^{d-1-k} - \binom{n}{k} x^{n-k} \right) (1-x)^{-(d-k)} \Bigg|_{x=2} .
\end{aligned}$$

Substituting in $x = 2$ at this point yields

$$\begin{aligned}
\sum_{k=0}^{n-d} \binom{k+d-1}{d-1} 2^k =& \sum_{k=0}^{d-1} \left(\binom{d-1}{k} 2^{d-1-k} - \binom{n}{k} 2^n 2^{-k} \right) (-1)^{d-k} \\
&= -\sum_{k=0}^{d-1} \binom{d-1}{k} (-2)^{d-1-k} - 2^n \sum_{k=0}^{d-1} \binom{n}{k} (-1)^{d-k} 2^{-k} \,.
\end{aligned}$$

Noting that $(x+y)^{d-1} = \sum_{k=0}^{d-1}\binom{d-1}{k}x^k y^{d-1-k}$ then substituting $x = 1$ and $y = -2$ we see that the left sum reduces to $(-1)^{d-1}$ yielding

$$\begin{aligned}\sum_{k=0}^{n-d}\binom{k+d-1}{d-1}2^k &= (-1)^d - 2^n\sum_{k=0}^{d-1}\binom{n}{d-1-k}(-1)^{k+1}2^{-(d-1-k)}\\ &= (-1)^d + 2^{n-d+1}\sum_{k=0}^{d-1}\binom{n}{d-1-k}(-2)^k\,,\end{aligned}$$

as required. □

If $n \geq 2(d-1)$ then this result can be bounded by $1 + 2^{n+1}\binom{n}{d-1}$. Following from this we have the asymptotic estimate

$$1 + 2^{n+1}\frac{n!}{(n-d+1)!(d-1)!} = 2^{n+1}\left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2})\right)$$

which we see grows asymptotically like $\mathcal{O}\left(2^n n^{d-1}\right)$ which is much slower than the $\mathcal{O}\left(2^{dn}\right)$ for the full grid ($d \geq 2$).

The calculation for a sparse grid with boundary points is more complex and is not something I have come across in the literature.

Proposition 2.18. *For a level $n \geq 2(d-1)$ sparse grid with boundary points the number of unknowns is bounded above by*

$$1 + 2^{n-d+1}(6^d - 5^d)\binom{n}{d-1}\,. \tag{2.6}$$

Proof. We first observe that on the boundaries lie level n sparse grids of smaller dimensions, e.g. the $(d-1)$-faces on a d-cube are sparse grids of dimension $d-1$ without boundaries, the $(d-2)$-faces are sparse grids of dimension $d-2$ without boundaries, etc. until one gets to the 2^d vertices in the corners. Given that there are $2^{d-e}\binom{d}{e}$ e-faces on a d-cube we apply (2.5) to find that the number of unknowns for a sparse grid with boundaries to be

$$\begin{aligned}&2^d + \sum_{e=1}^{d}2^{d-e}\binom{d}{e}\left((-1)^e + 2^{n-e+1}\sum_{k=0}^{e-1}\binom{n}{e-1-k}(-2)^k\right)\\ =&1 + 2^{n-d+1}\sum_{e=1}^{d}2^{2(d-e)}\binom{d}{e}\sum_{k=0}^{e-1}\binom{n}{e-1-k}(-2)^k\,. \end{aligned} \tag{2.7}$$

Given $n \geq 2(d-1)$ then $\binom{n}{e-1-k} \leq \binom{n}{d-1}$ for all $e = 1, \dots, d$ and $k = 0, \dots, e-1$, thus

$$\begin{aligned}
(2.7) &\leq 1 + 2^{n-d+1} \sum_{e=1}^{d} 2^{2(d-e)} \binom{d}{e} \sum_{k=0}^{e-1} \binom{n}{d-1} 2^k \\
&= 1 + 2^{n-d+1} \binom{n}{d-1} \sum_{e=1}^{d} 4^{d-e}(2^e - 1) \binom{d}{e} \\
&= 1 + 2^{n-d+1} \binom{n}{d-1} 4^d \sum_{e=0}^{d} (2^{-e} - 4^{-e}) \binom{d}{e} \\
&= 1 + 2^{n-d+1} \binom{n}{d-1} 4^d \left((3/2)^d - (5/4)^d \right) \\
&= 1 + 2^{n-d+1} (6^d - 5^d) \binom{n}{d-1} ,
\end{aligned}$$

as required. □

We see that the asymptotic rate with respect to n is the same as that without boundaries, namely $\mathcal{O}(2^n n^{d-1})$. For $d = 1$ we can see the result is $1 + 2^n$ which is exactly the number of unknowns in the level n discretisation of a unit interval with boundaries. However, for $d > 1$ this result over-estimates of the number of unknowns. In practice one can use (2.7) to quickly compute the exact number of unknowns in a sparse grid with boundaries.

2.1.3 Error of sparse grid interpolation

Now we consider the error for interpolation $u \in H^2_{\text{mix}}$ onto a sparse grid. The result will be obtained by bounding the contribution from hierarchical spaces that are not included in the sparse grid. This is done using the result of Lemma 2.15, namely for $\underline{l} \geq \underline{1}$

$$\left\| u^h_{\underline{l}} \right\|_2 \leq \left(\frac{1}{3} \right)^d 2^{-2|\underline{l}|} \| D^{\underline{2}} u \|_2 \, .$$

We start by deriving the classical result for functions which are zero on the boundary of the domain, that is $u \in H^2_{0,\text{mix}}$ and later exploit the fact that the boundary is made up of lower dimensional sparse grids. Note that this result again differs slightly from the literature due to the different convention of only including hierarchical spaces up to level n in the sparse grid (as opposed to $n + d - 1$).

Proposition 2.19 ([22, 50])**.** *Let $u_n^s = \sum_{|\underline{l}|\le n\,\&\,\underline{l}\ge\underline{1}} u_{\underline{l}}^h$ be the level n sparse grid interpolant of $u \in H^2_{0,mix}([0,1]^d)$, then*

$$\|u - u_n^s\|_2 \le 2^{-2n}\frac{1}{3}\left(\frac{1}{3}\right)^d \|D^{\underline{2}}u\|_2 \sum_{k=0}^{d-1}\binom{n}{k}\left(\frac{1}{3}\right)^{d-1-k}. \qquad (2.8)$$

Proof. Given the hierarchical decomposition of u and u_n^s it follows that the error of sparse grid interpolation is

$$\begin{aligned}\|u-u_n^s\|_2 &\le \sum_{|\underline{l}|>n\,\&\,\underline{l}\ge\underline{1}} \|u_{\underline{l}}^h\|_2\\ &\le \sum_{|\underline{l}|>n\,\&\,\underline{l}\ge\underline{1}} \left(\frac{1}{3}\right)^d 2^{-2|\underline{l}|}\|D^{\underline{2}}u\|_2\\ &= \left(\frac{1}{3}\right)^d \|D^{\underline{2}}u\|_2 \sum_{k=n+1}^{\infty}\sum_{|\underline{l}|=k\,\&\,\underline{l}\ge\underline{1}} 2^{-2k}.\end{aligned}$$

As the number of $\underline{l}$ satisfying $|\underline{l}| = k\,\&\,\underline{l}\ge\underline{1}$ is equal to $\binom{k-1}{d-1}$ one has

$$\begin{aligned}\|u-u_n^s\|_2 &\le \left(\frac{1}{3}\right)^d \|D^{\underline{2}}u\|_2 \sum_{k=n+1}^{\infty}\binom{k-1}{d-1}2^{-2k}\\ &= 2^{-2(n+1)}\left(\frac{1}{3}\right)^d \|D^{\underline{2}}u\|_2 \sum_{k=0}^{\infty}\binom{k+n}{d-1}2^{-2k}.\end{aligned}$$

For the sum we can use a similar approach as used in Proposition 2.17. Let $0 < x < 1$, then

$$\begin{aligned}\sum_{k=0}^{\infty} x^k\binom{k+n}{d-1} &= \frac{x^{-n}}{(d-1)!}\left(\sum_{k=0}^{\infty}x^{k+n}\right)^{(d-1)}\\ &= \frac{x^{-n}}{(d-1)!}\left(\frac{x^n}{1-x}\right)^{(d-1)}\\ &= \frac{x^{-n}}{(d-1)!}\sum_{k=0}^{d-1}\binom{d-1}{k}(x^n)^{(k)}\left(\frac{1}{1-x}\right)^{(d-1-k)}\\ &= \frac{x^{-n}}{(d-1)!}\sum_{k=0}^{d-1}\binom{d-1}{k}\frac{x^{n-k}n!}{(n-k)!}\frac{(d-1-k)!}{(1-x)^{d-k}}\\ &= \sum_{k=0}^{d-1}\binom{n}{k}\left(\frac{x}{1-x}\right)^{d-1-k}\frac{1}{1-x}.\end{aligned}$$

Substituting $x = 2^{-2}$ one obtains

$$\sum_{k=0}^{d-1}\binom{n}{k}\left(\frac{1}{3}\right)^{d-1-k}\frac{4}{3}$$

therefore yielding the error bound

$$\|u - u_n^s\|_2 \leq 2^{-2n}\frac{1}{3}\left(\frac{1}{3}\right)^d \|D^{\underline{2}}u\|_2 \sum_{k=0}^{d-1}\binom{n}{k}\left(\frac{1}{3}\right)^{d-1-k}$$

as desired. □

We note that if $n \geq 2(d-1)$ we can sacrifice some tightness of the bound for simplicity to obtain

$$\|u - u_n^s\|_2 \leq 2^{-2n}\frac{1}{2}\left(\frac{1}{3}\right)^d \|D^{\underline{2}}u\|_2 \binom{n}{d-1} . \tag{2.9}$$

Thus one expects an asymptotic rate of convergence of $\mathcal{O}(2^{-2n}n^{d-1})$ with respect to n.

For comparison we can apply the same technique to estimate the error for interpolation onto a full grid $u_{\underline{i}}$. This has been done for an isotropic full grid $u_{(n,\dots,n)}$ (see [22]) which we extend here to an anisotropic full grid.

Lemma 2.20. *Let $u \in H^2_{0,mix}$ and $\underline{i} \in \mathbb{N}^d$, then the piecewise multi-linear interpolant $u_{\underline{i}} \in V_{\underline{i}}$ of u satisfies*

$$\|u - u_{\underline{i}}\|_2 \leq 9^{-d}\|D^{\underline{2}}u\|_2 \sum_{k=1}^{d} 4^{-i_k} . \tag{2.10}$$

Proof.

$$\begin{aligned}
\|u - u_{\underline{i}}\|_2 &\leq \sum_{\underline{l}\not\leq\underline{i}} \|u_{\underline{l}}^h\|_2 \\
&\leq 3^{-d}\|D^{\underline{2}}u\|_2 \sum_{\underline{l}\not\leq\underline{i}} 2^{-2|\underline{l}|} \\
&= 3^{-d}\|D^{\underline{2}}u\|_2 \left(\sum_{\underline{l}\geq\underline{1}} 2^{-2|\underline{l}|} - \sum_{\underline{1}\leq\underline{l}\leq\underline{i}} 2^{-2|\underline{l}|}\right) \\
&= 3^{-d}\|D^{\underline{2}}u\|_2 \left(\left(\sum_{l=1}^{\infty} 4^{-l}\right)^d - \prod_{k=1}^{d}\left(\sum_{l=1}^{i_k} 4^{-l}\right)\right) \\
&= 9^{-d}\|D^{\underline{2}}u\|_2 \left(1 - \prod_{k=1}^{d}(1 - 4^{-i_k})\right) .
\end{aligned}$$

Now we claim that given $x_1, \dots, x_d \in (0,1)$ one has

$$\sum_{k=1}^{d} x_k \geq 1 - \prod_{k=1}^{d}(1 - x_k).$$

If $d = 1$ then one has $x_1 = 1-(1-x_1)$ and for $d = 2$ one has $1-(1-x_1)(1-x_2) = x_1 + x_2 - x_1x_2 \leq x_1 + x_2$. Assuming the claim is true for the case $d-1$, then for the case d one has

$$\begin{aligned} 1-\prod_{k=1}^{d}(1-x_k) &= 1-\prod_{k=1}^{d-1}(1-x_k) + x_d\prod_{k=1}^{d-1}(1-x_k) \\ &\leq \sum_{k=1}^{d-1} x_k + x_d = \sum_{k=1}^{d} x_k \,, \end{aligned}$$

and thus the claim is true for all d by induction. It follows that

$$\|u-u_{\underline{i}}\|_2 \leq 9^{-d}\|D^{\underline{2}}u\|_2 \sum_{k=1}^{d} 4^{-i_k}$$

as required. □

Note that it can be similarly shown [22] that

$$\|u-u_{\underline{i}}\|_\infty \leq 6^{-d}\|D^{\underline{2}}u\|_\infty \sum_{k=1}^{d} 4^{-i_k} \,.$$

If $\underline{i} = (n,\dots,n)$ for some $n \in \mathbb{N}$ then our bound (2.10) reduces to the known result for an isotropic grid $\|u-u_{(n,\dots,n)}\|_2 \leq d\,9^{-d}4^{-n}\|D^{\underline{2}}u\|_2$. Here we see the expected convergence rate of $\mathcal{O}(4^{-n})$ for isotropic full grids in which case it is clear that the sparse grid is only a factor of n^{d-1} slower to converge asymptotically with respect to n despite having $\mathcal{O}(n^{d-1}2^{-n(d-1)})$ times the number of unknowns. Figure 2.3 depicts the difference in rates of convergence for the right-hand sides of (2.9) and (2.10) with respect to the number of unknowns.

We pause for a moment to prove an inequality stronger than Friedrich's inequality [32] for functions $u \in H^2_{0,\mathrm{mix}}([0,1]^2)$. As far as I am aware this has not been shown in any other literature.

Lemma 2.21. *If $u \in H^2_{0,mix}([0,1]^2)$ then*

$$\|u\|_2 \leq 9^{-d}\|D^{\underline{2}}u\|_2 \,. \tag{2.11}$$

Proof. We note that as u can be expressed in terms of the a sum of its hierarchical components $u = \sum_{\underline{l}\geq\underline{1}} u^h_{\underline{l}}$. Therefore

$$\begin{aligned} \|u\|_2 \leq \sum_{\underline{l}\geq\underline{1}} \|u^h_{\underline{l}}\|_2 &\leq 3^{-d}\|D^{\underline{2}}u\|_2 \sum_{\underline{l}\geq\underline{1}} 2^{-2|\underline{l}|} \\ &= 3^{-d}\|D^{\underline{2}}u\|_2 \left(\sum_{l=1}^{\infty} 4^{-l}\right)^d = 9^{-d}\|D^{\underline{2}}u\|_2 \end{aligned}$$

as required. □

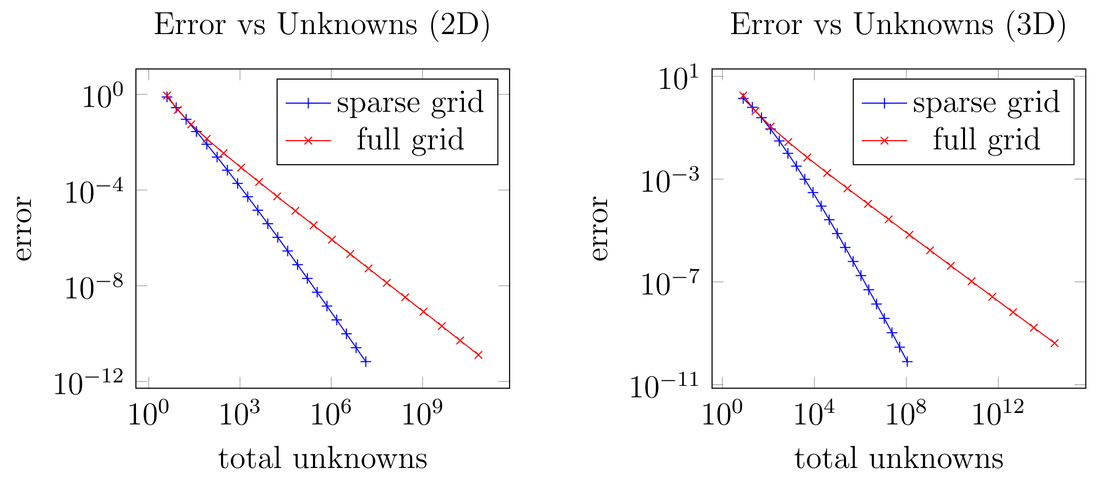


Figure 2.3: *We plot an example of the error against the number of unknowns in two and three dimensions. It is clear that a sparse grid can achieve the same accuracy as a full grid with far fewer unknowns.*

We now move on to the case with non-zero boundary. As far as I know this has not been covered in the existing literature. When we estimated the unknowns on the sparse grid with boundaries we noted that the boundaries are composed of sparse grids in lower dimensions without boundaries plus the 2^d vertices of the d-cube. This same property may be used in the analysis of sparse grid interpolation for functions $u \in H^2_{\text{mix}}$. We first introduce some notation. Given $\Omega = [0,1]^d$ we let $\partial\Omega$ denote the boundary of Ω, that is the union of the $2d$ $(d-1)$-faces which are each isomorphic to $[0,1]^{d-1}$. Similarly, for $1 < k \leq d$ we denote $\partial^k\Omega$ be the union of the $2^k\binom{d}{k}$ $(d-k)$-faces which are each isomorphic to $[0,1]^{d-k}$. Given a function $u \in H^2_{\text{mix}}$ then $|u|_{H^2_{\text{mix}}(\partial^k\Omega)}$ denotes the sum of the H^2_{mix} semi-norm over each of the $(d-k)$-faces of Ω, that is

$$|u|_{H^2_{\text{mix}}(\partial^k\Omega)} = \sum_{\{m_1,\dots,m_k\}\subset\{1,\dots,d\}} \sum_{y\in\{0,1\}^k} \|D^2 u|_{x_{m_1}=y_1,\dots,x_{m_k}=y_k}\|_{L_2([0,1]^{d-k})} \, .$$

Proposition 2.22. *Let $u \in H^2_{mix}([0,1]^d)$ and u^s_n be a level $n \geq 0$ sparse grid interpolant of u (including boundary points), then*

$$\|u - u^s_n\|_2 \leq \frac{1}{3} 2^{-2n} \sum_{k=1}^{d} 3^{-(d+k)/2} |u|_{H^2_{mix}(\partial^{d-k}\Omega)} \sum_{m=0}^{k-1} \binom{n}{m} \left(\frac{1}{3}\right)^{k-1-m} . \qquad (2.12)$$

Proof. Given the hierarchical decomposition of u and u_n^s (with $n \geq 0$) we have

$$\begin{aligned}\|u - u_n^s\|_2 &\leq \sum_{|\underline{i}|>n} \|u_{\underline{i}}^h\|_2 \\ &= \sum_{|\underline{i}|>n \,\&\, \underline{i}\geq\underline{1}} \|u_{\underline{i}}^h\|_2 + \sum_{|\underline{i}|>n \,\&\, \underline{i}\ngeq\underline{1}} \|u_{\underline{i}}^h\|_2 \,.\end{aligned}$$

The left sum corresponds to the interior points of the sparse grid and we may apply Proposition 2.19. We now concentrate on the right-hand sum. Note that $\underline{i} \ngeq \underline{1}$ is not the same as $\underline{i} < \underline{1}$, it is equivalent to the statement $\exists k \in \{1, \dots, d\}$ such that $i_k = 0$. The sum may be split up according to how many components of $\underline{i}$ are zero. If $\underline{i}$ has k zero components then $u_{\underline{i}}^h$ is determined by the values of u on $\partial^k\Omega$. With $|\underline{i}| > n \geq 0$ at least one component of $\underline{i}$ is always non-zero. Similarly, as $\underline{i} \ngeq \underline{1}$ at least one component will always be zero, thus $1 \leq k \leq d-1$. Hence we write

$$\sum_{|\underline{i}|>n \,\&\, \underline{i}\ngeq\underline{1}} \|u_{\underline{i}}^h\|_2 = \sum_{k=1}^{d-1} \sum_{|\underline{i}|_0=d-k \,\&\, |\underline{i}|>n} \|u_{\underline{i}}^h\|_2 \,.$$

For a given k, the sum $\sum_{|\underline{i}|_0=d-k\&|\underline{i}|>n} \|u_{\underline{i}}^h\|_2$ is over all nodal basis functions whose peak lies on the interior of the $2^k\binom{d}{k}$ $(d-k)$-faces making up $\partial^k\Omega$. This is precisely a $(d-k)$-dimensional sparse grid interpolation on these faces with a tensor product of $\phi_{0,0}$ or $\phi_{0,1}$ in the remaining k dimensions, in particular

$$\begin{aligned}\sum_{|\underline{i}|_0=d-k \,\&\, |\underline{i}|>n} \|u_{\underline{i}}^h\|_2 = &\sum_{\substack{\{m_1,\dots,m_k\}\cup\{m_{k+1},\dots,m_d\} \\ =\{1,\dots,d\}}} \sum_{y\in\{0,1\}^k} \\ &\sum_{\substack{i_{m_{k+1}}+\dots+i_{m_d}>n \\ i_{m_1}=\dots=i_{m_k}=0}} \left\| \left(u_{\underline{i}}^h\big|_{\substack{x_{m_1}=y_1, \\ \dots,x_{m_k}=y_k}} \right) \phi_{0,y_1}(x_{m_1}) \cdots \phi_{0,y_k}(x_{m_k}) \right\|_{L_2(\Omega)} .\end{aligned}$$

Now as

$$\begin{aligned}&\left\| \left(u_{\underline{i}}^h\big|_{\substack{x_{m_1}=y_1, \\ \dots,x_{m_k}=y_k}} \right) \phi_{0,y_1}(x_{m_1}) \cdots \phi_{0,y_k}(x_{m_k}) \right\|_{L_2(\Omega)} \\ &\qquad = \left\| u_{\underline{i}}^h\big|_{\substack{x_{m_1}=y_1, \\ \dots,x_{m_k}=y_k}} \right\|_{L_2([0,1]^{d-k})} \|\phi_{0,y_1}(x_{m_1})\|_{L_2([0,1])} \cdots \|\phi_{0,y_k}(x_{m_k})\|_{L_2([0,1])}\end{aligned}$$

and each of the $\|\phi_{0,y}(x)\|_{L_2([0,1])} = 3^{-1/2}$ by Lemma 2.7 then

$$\begin{aligned}&\sum_{|\underline{i}|_0=d-k \,\&\, |\underline{i}|>n} \|u_{\underline{i}}^h\|_2 \\ &= 3^{-k/2} \sum_{\substack{\{m_1,\dots,m_k\}\cup\{m_{k+1},\dots,m_d\} \\ =\{1,\dots,d\}}} \sum_{y\in\{0,1\}^k} \sum_{\substack{i_{m_{k+1}}+\dots+i_{m_d}>n \\ i_{m_1}=\dots=i_{m_k}=0}} \left\| u_{\underline{i}}^h\big|_{\substack{x_{m_1}=y_1, \\ \dots,x_{m_k}=y_k}} \right\|_{L_2([0,1]^{d-k}} .\end{aligned}$$

Now the remaining sums simply bound the sparse grid error on the interior of the $(d-k)$-dimensional boundaries for which Proposition 2.19 may be applied. It follows that

$$\begin{aligned}\sum_{|\underline{i}|>n \& \underline{i} \not> \underline{1}} \|u_{\underline{i}}^h\|_2 &\leq \sum_{k=1}^{d-1} 3^{-k/2} 2^{-2n} \frac{1}{3} 3^{-(d-k)} |u|_{H^2_{\text{mix}}(\partial^k \Omega)} \sum_{m=0}^{d-k-1} \binom{n}{m} \left(\frac{1}{3}\right)^{d-k-1-m} \\ &= \frac{1}{3} 2^{-2n} \sum_{k=1}^{d-1} 3^{-(d+k)/2} |u|_{H^2_{\text{mix}}(\partial^{d-k}\Omega)} \sum_{m=0}^{k-1} \binom{n}{m} \left(\frac{1}{3}\right)^{k-1-m} .\end{aligned}$$

Adding in the term for the interior of Ω one obtains

$$\|u-u_n^s\|_2 \leq \frac{1}{3} 2^{-2n} \sum_{k=1}^{d} 3^{-(d+k)/2} |u|_{H^2_{\text{mix}}(\partial^{d-k}\Omega)} \sum_{m=0}^{k-1} \binom{n}{m} \left(\frac{1}{3}\right)^{k-1-m} ,$$

as required. □

We can sacrifice the tightness of this bound to obtain something a little simpler for $n \geq 2(d-1)$, namely

$$\begin{aligned}\|u-u_n^s\|_2 &\leq \frac{1}{2} 2^{-2n} \sum_{k=1}^{d} 3^{-(d+k)/2} |u|_{H^2_{\text{mix}}(\partial^{d-k}\Omega)} \binom{n}{k-1} \\ &\leq \frac{1}{2(\sqrt{3}-1)} 3^{-d/2} 2^{-2n} \binom{n}{d-1} \sum_{k=1}^{d} |u|_{H^2_{\text{mix}}(\partial^{d-k}\Omega)} .\end{aligned}$$

One can also obtain an inequality similar to that of Lemma 2.21 in the case of a non-zero boundary. Taking (2.12) for $n=0$ and adding the level $\underline{0}$ hierarchical basis functions which are each 1 at one of the 2^d vertices one obtains

$$\|u\|_2 \leq 3^{-d/2} \left(\sum_{\underline{0} \leq \underline{j} \leq \underline{1}} |u(x_{\underline{0},\underline{j}})| \right) + \left(\sum_{k=1}^{d} 3^{-(d+3k)/2} |u|_{H^2_{\text{mix}}(\partial^{d-k}\Omega)} \right) .$$

2.1.4 Additional remarks

Classical sparse grids may not be suitable for all problems. For example, for some problems it may be desirable to solve some dimensions at a finer scale than others, e.g. consider phase space in which the velocity coordinates may not need be as fine as the spatial coordinates. Problems with multi-scale phenomena may also be problematic due to the difficulty in resolving fine scale effects in the absence of the finer hierarchical surpluses. Fortunately there are many adaptations and generalisations of classical sparse grids that address such issues. A particularly nice use of sparse grids is in multi-grid and multilevel methods [59, 19, 57]. There are also extensions which utilise higher order hierarchical basis functions, see for example [18]. Whilst finite difference methods for hyperbolic PDEs have been developed directly on the sparse grid, see for example [63, 60], this thesis will focus on the combination technique for approximating sparse grid solutions which is introduced in the following section.

2.2 The Combination Technique

The combination technique was introduced in [61] as a method of approximating sparse grid solutions without the need to compute directly with hierarchical basis functions. By adding approximations from several different $V_{\underline{i}}$ (which may be computed using nodal basis functions for example) one is able to obtain an approximation in V_n^s. We motivate the combination technique by discussing the problem of adding approximations from different approximation spaces in order to obtain an approximation in a larger space. After giving the classical combination formula we then provide formula for the total number of unknowns in 2 and 3 dimensions and a bound for higher dimensions. We then review estimates of the error of solutions obtained by the combination technique using an error model for known as an error splitting for the approximations in each of the $V_{\underline{i}}$.

2.2.1 Preliminaries and motivation

Suppose we have two approximations $u_{\underline{i}}, u_{\underline{j}}$ to the function $u \in H^2_{\text{mix}}$ in the function spaces $V_{\underline{i}}, V_{\underline{j}}$ respectively. In particular we have

$$u_{\underline{i}} \in \bigoplus_{\underline{k} \leq \underline{i}} W_{\underline{k}}\,, \quad u_{\underline{j}} \in \bigoplus_{\underline{k} \leq \underline{j}} W_{\underline{k}}\,.$$

Now consider the function $u_{\underline{i}} + u_{\underline{j}}$. Clearly this will give an approximation in the space

$$u_{\underline{i}} + u_{\underline{j}} \in V_{\underline{i}} + V_{\underline{j}} = \bigoplus_{\{\underline{k} | \underline{k} \leq \underline{i} \text{ or } \underline{k} \leq \underline{j}\}} W_{\underline{k}}\,.$$

Since $V_{\underline{i}}, V_{\underline{j}} \subset V_{\underline{i}} + V_{\underline{j}}$ then one expects that one should be able to obtain a better approximation to u in the space $V_{\underline{i}} + V_{\underline{j}}$. However, it is clear that $u_{\underline{i}} + u_{\underline{j}}$ will generally not be a good approximation to u, but typically a closer approximation to $2u$ (e.g. consider $\underline{i} = \underline{j}$). We can see what is happening if we look at the hierarchical decomposition of our approximations. We have

$$u_{\underline{i}} + u_{\underline{j}} = \left(\sum_{\underline{k} \leq \underline{i}} u^h_{\underline{k}}\right) + \left(\sum_{\underline{k} \leq \underline{j}} u^h_{\underline{k}}\right) = \left(\sum_{\{\underline{k} | \underline{k} \leq \underline{i} \text{ or } \underline{k} \leq \underline{j}\}} u^h_{\underline{k}}\right) + \left(\sum_{\underline{k} \leq \underline{i} \wedge \underline{j}} u^h_{\underline{k}}\right)\,.$$

From here it is clear that we have additional contributions to the hierarchical components $u^h_{\underline{k}}$ for $\underline{k} \leq \underline{i} \wedge \underline{j}$ which is equivalent to an additional contribution from $u_{\underline{i} \wedge \underline{j}}$. Therefore, in order to approximate u in $V_{\underline{i}} + V_{\underline{j}}$ we may compute

$$u_{\underline{i}} + u_{\underline{j}} - u_{\underline{i} \wedge \underline{j}}\,.$$

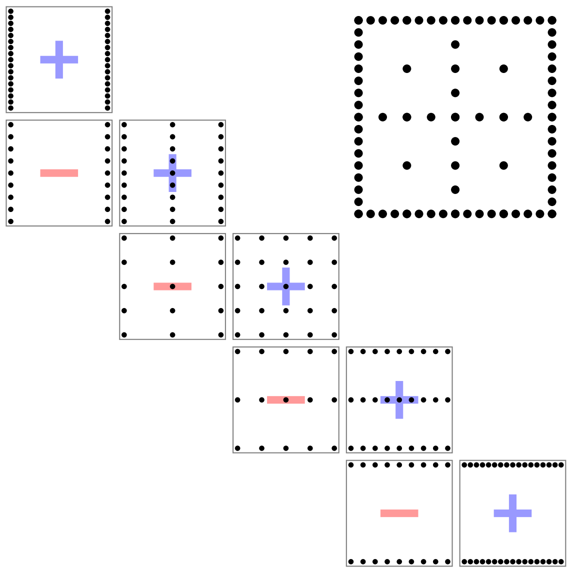


Figure 2.4: *The combination u_4^c for $d = 2$. The 9 component grids are arranged according to the frequency of data points in each dimension. A plus or minus denotes a combination coefficient of $+1$ or -1 respectively. In the top right is the (enlarged) sparse grid corresponding to the union of the component grids.*

This is effectively a simple application of the inclusion-exclusion principle, we add an approximation from two function spaces and subtract an approximation from the intersecting space to obtain an approximation in the combined space (or join space).

The combination technique can be viewed as the extension of this observation to obtain approximations in the sparse grid space V_n^s by attempting to add approximations from $V_{\underline{i}}$ for $|\underline{i}| = n$. The classical combination formula [50] is

$$u_n^c = \sum_{k=0}^{d-1} (-1)^k \binom{d-1}{k} \sum_{|\underline{l}|=n-k} u_{\underline{l}} \,, \tag{2.13}$$

where u_n^c is referred to as the combination solution and the $u_{\underline{i}} \in V_{\underline{i}}$ are often referred to as component solutions. A detailed derivation of this formula is given in Lemma 4.25 via the more general theory of adaptive sparse grids. An example of the combination technique in two spatial dimensions is depicted in Figure 2.4

For interpolation, that is $u_{\underline{i}} = \mathcal{I}_{\underline{i}} u$, the combination u_n^c is equal to the sparse grid interpolant [61, 50]. One can see this by decomposing each $u_{\underline{i}}$ into their hierarchical components which then cancel out (via the inclusion-exclusion principle as in our simple example) leaving the same hierarchical contributions as the sparse grid interpolant. As an immediate consequence the error bounds of

Propositions 2.19 and 2.22 may be applied to u_n^c when $u_{\underline{i}} = \mathcal{I}_{\underline{i}} u$.

2.2.2 Unknowns in the combination technique

Given that the $u_{\underline{i}}$ making up u_n^c have hierarchical components in common with each other it is clear that the combination technique has more unknowns than the sparse grid. Note that, whilst both u_n^s and u_n^c have range V_n^s, for the combination technique we count the total number of unknowns in the collection of function spaces $\{V_{\underline{i}} : n - d + 1 \leq |\underline{i}| \leq n\}$. The reason for this is that one needs to compute an approximation $u_{\underline{i}}$ in each $V_{\underline{i}}$ to construct u_n^c and as this is generally done for each $u_{\underline{i}}$ independently (despite some redundancy in the information they carry) we sum the number of unknowns in each $V_{\underline{i}}$ for which $u_{\underline{i}}$ is computed. The obvious question that arises is whether the redundancies are few enough that the combination technique is still an efficient way to compute approximations of u in V_n^s. If an approximation $u_{\underline{i}}$ includes boundary points then the number of unknowns is

$$\prod_{k=1}^{d} (1 + 2^{i_k}) \, .$$

We denote the sum over the unknowns in each $u_{\underline{i}}$ with $|\underline{i}| = n$ (with $\underline{i} \in \mathbb{N}^d$) by

$$\alpha_{n,d} := \sum_{\{\underline{i} \in \mathbb{N}^d : |\underline{i}| = n\}} \prod_{k=1}^{d} (1 + 2^{i_k}) \, . \tag{2.14}$$

Therefore the total number of unknowns computed for the combination u_n^c is $\sum_{k=0}^{d-1} \alpha_{n-k,d}$. For $d = 2$ we have the following result.

Lemma 2.23. *Given $\alpha_{n,d}$ defined by (2.14) and $d = 2$ then*

$$\alpha_{n,2} = n - 1 + (n + 5) 2^n$$

and therefore the total number of unknowns in the computation of u_n^c is

$$2n - 3 + \left(\frac{3}{2} n + 7 \right) 2^n \, .$$

Proof. We have

$$\begin{aligned}
\alpha_{n,2} = \sum_{i_1+i_2=n} (1+2^{i_1})(1+2^{i_2}) &= \sum_{i_1=0}^{n} (1+2^{i_1})(1+2^{n-i_1}) \\
&= \sum_{i_1=0}^{n} (1+2^{i_1}+2^{n-i_1}+2^n) \\
&= (n+1)(1+2^n) + 2(2^{n+1}-1) \\
&= n-1+(n+5)2^n\,.
\end{aligned}$$

It follows that

$$\begin{aligned}
\alpha_{n,2}+\alpha_{n-1,2} &= n-1+(n+5)2^n+n-2+(n+4)2^{n-1} \\
&= 2n-3+\left(\frac{3}{2}n+7\right)2^n\,,
\end{aligned}$$

as required. □

Notice that $\alpha_{n,d}$ may be calculated recursively as for $d \geq 2$ one has

$$\alpha_{n,d} = \sum_{i_d=0}^{n} \left(\sum_{i_1+\cdots+i_{d-1}=n-i_d} \prod_{k=1}^{d} (1+2^{i_k}) \right) = \sum_{i_d=0}^{n} (1+2^{i_d})\alpha_{n-i_d,d-1}\,. \tag{2.15}$$

This property will be useful in the following result for $d = 3$.

Lemma 2.24. *Given $\alpha_{n,d}$ defined by (2.14) and $d = 3$ then*

$$\alpha_{n,3} = \frac{1}{2}n^2 - \frac{3}{2}n - 5 + \left(\frac{1}{2}n^2 + \frac{15}{2}n + 13\right)2^n$$

and therefore the total number of unknowns in the computation of u_n^c is

$$\frac{3}{2}n^2 - \frac{15}{2}n - 8 + \left(\frac{7}{8}n^2 - \frac{97}{8}n + 16\right)2^n\,.$$

Proof. Using the recursion (2.15) and Lemma 2.23 we have

$$\alpha_{n,3} = \sum_{i_3=0}^{n} (1+2^{i_3})\alpha_{n,2} = \sum_{i_3=0}^{n} (1+2^{i_3})\left(n-i_3-1+(n-i_3+5)2^{n-i_3}\right)\,.$$

Note that via the substitution $i_3 \mapsto n-i_3$ one has $\sum_{i_3=0}^{n} i_3 2^{i_3} = \sum_{i_3=0}^{n}(n-i_3)2^{n-i_3}$. Therefore

$$\begin{aligned}
\alpha_{n,3} &= \sum_{i_3=0}^{n} n - i_3 - 1 + (n-i_3+5)2^{n-i_3} + (n-i_3-1)2^{i_3} + (n-i_3+5)2^n \\
&= \sum_{i_3=0}^{n} n - i_3 - 1 + 5\cdot 2^{n-i_3} + (n-1)2^{i_3} + (n-i_3+5)2^n \\
&= \sum_{i_3=0}^{n} (n-1+(n+5)2^n) - (1+2^n)\, i_3 + 5\cdot 2^{n-i_3} + (n-1)2^{i_3} \\
&= (n-1+(n+5)2^n)(n+1) - (1+2^n)\frac{n(n+1)}{2} + (n+4)\left(2^{n+1}-1\right) \\
&= \frac{1}{2}n^2 - \frac{3}{2}n - 5 + \left(\frac{1}{2}n^2 + \frac{15}{2}n + 13\right)2^n .
\end{aligned}$$

From here it is straightforward to show

$$\alpha_{n,3} + \alpha_{n-1,3} + \alpha_{n-2,3} = \frac{3}{2}n^2 - \frac{15}{2}n - 8 + \left(\frac{7}{8}n^2 - \frac{97}{8}n + 16\right)2^n ,$$

as required. □

Evaluating this sum for a general number of dimensions unfortunately does not reduce to a simple equation. However, it may be shown that $\alpha_{n,d} = \frac{n^{d-1}}{(d-1)!}2^n + \mathcal{O}(n^{d-2}2^n)$. Note that

$$2^{|\underline{i}|} = \prod_{t=1}^{d} 2^{i_t} \le \prod_{t=1}^{d}(2^{i_t}+1) \le \prod_{t=1}^{d} 2^{1+i_t} = 2^{d+|\underline{i}|} \tag{2.16}$$

which leads to the lower and upper bounds

$$\alpha_{n,d} \ge \binom{n+d-1}{d-1}2^n , \qquad \alpha_{n,d} \le \binom{n+d-1}{d-1}2^{d+n}$$

respectively. Thus asymptotically the total number of unknowns in computing u_n^c with respect to n is

$$\sum_{k=0}^{d-1}\binom{n-k+d-1}{d-1}2^{n-k} = \frac{n^{d-1}}{(d-1)!}2^{n+1}(1-2^{-d}) + \mathcal{O}\left(n^{d-2}2^n\right)$$

in which case we see the asymptotic growth rate of unknowns in the combination technique is the same as that for sparse grids up to a constant.

Summarising so far, we have that the combination technique has a similar number of unknowns compared to sparse grid and gives the same result for piecewise linear interpolation of $u \in H^2_{\text{mix}}$. We will now look at error estimates of u_n^c for $u_{\underline{i}}$ which are not necessarily piecewise linear interpolants of u.

2.2.3 Error of the combination technique

A classical result in [59] for the error of combination technique solutions u_n^c is obtained by assuming an error model for the $u_{\underline{i}}$ which is known as an error splitting. For 2 dimensional problems, an approximation $u_{\underline{i}}$ of u satisfies the error splitting model if the pointwise error expansion

$$u - u_{\underline{i}} = C_1(h_{i_1})h_{i_1}^p + C_2(h_{i_2})h_{i_2}^p + D(h_{i_1}, h_{i_2})h_{i_1}^p h_{i_2}^p \tag{2.17}$$

holds where $h_k := 2^{-k}$, $p > 0$ and C_1, C_2, D implicitly depend on $x \in \Omega$. Using this error model, the following Lemma bounds the error of $|u - u_n^c|$ which is similar to the result in [59] but with our slightly different definition of u_n^c and arbitrary $p > 0$ (rather than just $p = 2$).

Lemma 2.25 (Adapted from [61]). *Let $u_{\underline{i}} : [0,1]^2 \mapsto \mathbb{R}$ for $\underline{i} \in \mathbb{N}^2$ be approximations to some function $u : [0,1]^2 \mapsto \mathbb{R}$ which satisfies the pointwise error expansion* (2.17) *with $p > 0$ and $|C_1(h_{i_1})| \leq K$, $|C_2(h_{i_2})| \leq K$ and $|D(h_{i_1}, h_{i_2})| \leq K$ $\forall \underline{i}$ for some $K > 0$. Then the combination $u_n^c = \sum_{|\underline{i}|=n} u_{\underline{i}} - \sum_{|\underline{i}|=n-1} u_{\underline{i}}$ satisfies*

$$|u - u_n^c| \leq (3 + (1 + 2^p)n)\, K h_n^p \tag{2.18}$$

pointwise.

Proof. First notice that as $\sum_{|\underline{i}|=n} u - \sum_{|\underline{i}|=n-1} u = (n+1)u - nu = u$ one has

$$u - u_n^c = u - \left(\sum_{|\underline{i}|=n} u_{\underline{i}} - \sum_{|\underline{i}|=n-1} u_{\underline{i}} \right) = \sum_{|\underline{i}|=n} (u - u_{\underline{i}}) - \sum_{|\underline{i}|=n-1} (u - u_{\underline{i}}) .$$

Now substituting the error splitting (2.17) we obtain

$$\begin{aligned} u - u_n^c &= \sum_{|\underline{i}|=n} \left(C_1(h_{i_1})h_{i_1}^p + C_2(h_{i_2})h_{i_2}^p + D(h_{i_1}, h_{i_2})h_{i_1}^p h_{i_2}^p \right) - \\ &\qquad - \sum_{|\underline{i}|=n-1} \left(C_1(h_{i_1})h_{i_1}^p + C_2(h_{i_2})h_{i_2}^p + D(h_{i_1}, h_{i_2})h_{i_1}^p h_{i_2}^p \right) \\ &= \sum_{i_1=0}^{n} C_1(h_{i_1})h_{i_1}^p - \sum_{i_1=0}^{n-1} C_1(h_{i_1})h_{i_1}^p + \sum_{i_2=0}^{n} C_2(h_{i_2})h_{i_2}^p - \sum_{i_2=0}^{n-1} C_2(h_{i_2})h_{i_2}^p \\ &\quad + \sum_{i_1+i_2=n} D(h_{i_1}, h_{i_2})h_n^p - \sum_{i_1+i_2=n-1} D(h_{i_1}, h_{i_2})h_{n-1}^p \\ &= \left(C_1(h_n) + C_2(h_n) + \sum_{i_1+i_2=n} D(h_{i_1}, h_{i_2}) - 2^p \sum_{i_1+i_2=n-1} D(h_{i_1}, h_{i_2}) \right) h_n^p . \end{aligned}$$

Taking the absolute value of both sides we have

$$\begin{aligned}|u-u_n^c| &\leq |C_1(h_n)|h_n^p+|C_2(h_n)|h_n^p \\ &\quad +\left(\sum_{i_1+i_2=n}|D(h_{i_1},h_{i_2})|+2^p\sum_{i_1+i_2=n-1}|D(h_{i_1},h_{i_2})|\right)h_n^p \\ &\leq Kh_n^p+Kh_n^p+(n+1)Kh_n^p+2^pnKh_n^p \\ &= (3+(1+2^p)n)\,Kh_n^p\,,\end{aligned}$$

as required. □

A similar result was also derived in [59] for $d=3$ dimensions assuming the pointwise error splitting

$$\begin{aligned}u-u_{\underline{i}} &= C_1(h_{i_1})h_{i_1}^p+C_2(h_{i_2})h_{i_2}^p+C_3(h_{i_3})h_{i_3}^p \\ &\quad +D_1(h_{i_1},h_{i_2})h_{i_1}^ph_{i_2}^p+D_2(h_{i_1},h_{i_3})h_{i_1}^ph_{i_3}^p+D_3(h_{i_2},h_{i_3})h_{i_2}^ph_{i_3}^p \\ &\quad +E(h_{i_1},h_{i_2},h_{i_3})h_{i_1}^ph_{i_2}^ph_{i_3}^p\,. \qquad (2.19)\end{aligned}$$

An analogous result to Lemma 2.25 holds in three dimensions.

Lemma 2.26 (Adapted from [61]). *Let $u_{\underline{i}}:[0,1]^3\mapsto\mathbb{R}$ for $\underline{i}\in\mathbb{N}^3$ be approximations to some function $u:[0,1]^3\mapsto\mathbb{R}$ satisfying the pointwise error splitting* (2.19) *with $|C_1(h_{i_1})|\leq K$, $|C_2(h_{i_2})|\leq K$, $|C_3(h_{i_3})|\leq K$, $|D_1(h_{i_1},h_{i_2})|\leq K$, $|D_2(h_{i_1},h_{i_3})|\leq K$, $|D_3(h_{i_2},h_{i_3})|\leq K$ and $|E(h_{i_1},h_{i_2},h_{i_3})|\leq K$ for all $\underline{i}$ for some $K>0$. Then the combination*

$$u_n^c=\sum_{|\underline{i}|=n}u_{\underline{i}}-2\sum_{|\underline{i}|=n-1}u_{\underline{i}}+\sum_{|\underline{i}|=n-2}u_{\underline{i}}$$

satisfies the pointwise bound

$$|u-u_n^c|\leq\left(7+\left(9+2^{3+p}-2^{2p}\right)\frac{n}{2}+(1+2^p)^2\frac{n^2}{2}\right)Kh_n^p\,. \qquad (2.20)$$

Proof. The approach is the same as that of Lemma 2.25 and the details essentially the same as those in [59]. We give a rough sketch of the proof for completeness. As $\binom{n+2}{2}-2\binom{n+1}{2}+\binom{n}{2}=1$ one has

$$u-u_n^c=\sum_{|\underline{i}|=n}(u-u_{\underline{i}})-2\sum_{|\underline{i}|=n-1}(u-u_{\underline{i}})+\sum_{|\underline{i}|=n-2}(u-u_{\underline{i}})\,.$$

One may now substitute the error splitting (2.19). We focus on each term individually. For the C_1 term by noting $\sum_{|\underline{i}|=n}f(i_1,i_2,i_3)=\sum_{i_1=0}^{n}\sum_{i_2=0}^{n-i_1}f(i_1,i_2,n-$

$i_1 - i_2$) one has

$$\begin{aligned}
&\sum_{|\underline{i}|=n} C_1(h_{i_1})h_{i_1}^p - 2\sum_{|\underline{i}|=n-1} C_1(h_{i_1})h_{i_1}^p + \sum_{|\underline{i}|=n-2} C_1(h_{i_1})h_{i_1}^p \\
=&\sum_{i_1=0}^{n}\sum_{i_2=0}^{n-i_1} C_1(h_{i_1})h_{i_1}^p - 2\sum_{i_1=0}^{n-1}\sum_{i_2=0}^{n-1-i_1} C_1(h_{i_1})h_{i_1}^p + \sum_{i_1=0}^{n-2}\sum_{i_2=0}^{n-2-i_1} C_1(h_{i_1})h_{i_1}^p \\
=&C_1(h_n)h_n^p + \sum_{i_1=0}^{n-1}\left(\sum_{i_2=0}^{n-i_1} C_1(h_{i_1})h_{i_1}^p - \sum_{i_2=0}^{n-1-i_1} C_1(h_{i_1})h_{i_1}^p\right) \\
&- C_1(h_{n-1})h_{n-1}^p - \sum_{i_1=0}^{n-2}\left(\sum_{i_2=0}^{n-1-i_1} C_1(h_{i_1})h_{i_1}^p - \sum_{i_2=0}^{n-2-i_1} C_1(h_{i_1})h_{i_1}^p\right) \\
=&C_1(h_n)h_n^p + \sum_{i_1=0}^{n-1} C_1(h_{i_1})h_{i_1}^p - C_1(h_{n-1})h_{n-1}^p - \sum_{i_1=0}^{n-2} C_1(h_{i_1})h_{i_1}^p = C_1(h_n)h_n^p \,.
\end{aligned}$$

The C_2, C_3 terms are similar. For the D_1 term one has

$$\sum_{|\underline{i}|=n} D_1(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p - 2\sum_{|\underline{i}|=n-1} D_1(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p + \sum_{|\underline{i}|=n-2} D_1(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p \,. \tag{2.21}$$

For the first sum one has

$$\begin{aligned}
\sum_{|\underline{i}|=n} D_1(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p &= \sum_{i_1=0}^{n}\sum_{i_2=0}^{n-i_1} D_1(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p \\
&= \sum_{i_1=0}^{n} D_1(h_{i_1},h_{n-i_1})h_n^p + \sum_{i_1=0}^{n-1}\sum_{i_2=0}^{n-1-i_1} D_1(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p \,.
\end{aligned}$$

Similarly for the second sum one has

$$\sum_{|\underline{i}|=n-1} D_1(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p = \sum_{i_1=0}^{n-1} D_1(h_{i_1},h_{n-1-i_1})h_{n-1}^p + \sum_{i_1=0}^{n-2}\sum_{i_2=0}^{n-2-i_1} D_1(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p \,.$$

Substituting these two equalities back into (2.21) therefore yields

$$\sum_{i_1=0}^{n} D_1(h_{i_1},h_{n-i_1})h_n^p - \sum_{i_1=0}^{n-1} D_1(h_{i_1},h_{n-1-i_1})h_{n-1}^p \,.$$

The D_2, D_3 terms are similar. There is no cancellation in the E terms so that one simply has

$$\sum_{|\underline{i}|=n} E(h_{i_1},h_{i_2},h_{i_3})h_n^p - 2\sum_{|\underline{i}|=n-1} E(h_{i_1},h_{i_2},h_{i_3})h_{n-1}^p + \sum_{|\underline{i}|=n-2} E(h_{i_1},h_{i_2},h_{i_3})h_{n-2}^p \,.$$

Thus, taking the absolute values, bounding the remaining C_k, D_k and E by K and summing the terms one obtains

$$\begin{aligned}|u - u_n^c| \leq 3Kh_n^p &+ 3\left(1 + (1+2^p)n\right) Kh_n^p \\ &+ \left(\frac{(n+2)(n+1)}{2} + 2^{p+1}\frac{(n+1)n}{2} + 2^{2p}\frac{n(n-1)}{2}\right) Kh_n^p\end{aligned}$$

from which the result follows. $\square$

We note that the asymptotic rates of convergence for Lemmas 2.25 and 2.26 with $p = 2$ are $\mathcal{O}(n^{d-1}2^{-2n})$ which is the same as the piecewise multi-linear interpolation error for sparse grids. It is noted [59] that this type of proof is easily extended to higher dimensions, however it is not immediately clear what the constants will be. Something else that isn't clear in these results is how large K may be. This will be problem dependant but if it also increases exponentially in the number of dimensions then larger n may be required in higher dimensions to obtain useful results.

Some of these questions are addressed by Reisinger [110, 109] who extends the two and three dimensional results to arbitrary dimensions. We will review this work but again, we derive slightly different results given our context of including boundary points and the different definition of u_n^c. Qualitatively the results will be the same but constants will differ.

Theorem 2.27 (Adapted from [110])**.** *For $\underline{i} \in \mathbb{N}^d$ let $u_{\underline{i}} : [0,1]^d \mapsto \mathbb{R}$ be an approximation to $u : [0,1]^d \mapsto \mathbb{R}$ satisfying the pointwise error expansion*

$$u - u_{\underline{i}} = \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} v_{j_1,\dots,j_m}(h_{i_{j_1}},\dots,h_{i_{j_m}}) h_{i_{j_1}}^p \cdots h_{i_{j_m}}^p \,. \tag{2.22}$$

Additionally, suppose there is some $K > 0$ such that

$$|v_{j_1,\dots,j_m}(h_{j_1},\dots,h_{j_m})| \leq K \qquad \forall 1 \leq m \leq d \text{ and } \forall \{j_1,\dots,j_d\} \subset \{1,\dots,d\}\,.$$

Then, the combination

$$u_n^c = \sum_{k=0}^{d-1} (-1)^k \binom{d-1}{k} \sum_{|\underline{i}|=n-k} u_{\underline{i}}$$

satisfies the pointwise error bound

$$|u - u_n^c| \leq K 2^{-pn}(1+2^p)^{d-1}\binom{n+2d-1}{d-1}\,.$$

Before we prove this we first introduce some notation and prove a few combinatorial identities. We define

$$S(n,d) = \sum_{\{\underline{i}\in\mathbb{N}^d : |\underline{i}|=n\}} u_{\underline{i}}$$

where we will typically omit $\underline{i} \in \mathbb{N}^d$. We also define the difference operator δ on functions $f : \mathbb{N} \mapsto \mathbb{R}$ by

$$\delta f(n) := f(n) - f(n-1)\,.$$

Lemma 2.28 ([110]). *The combination formula in d dimensions is given by*

$$\delta^{d-1}S(n,d) = \sum_{k=0}^{d-1}(-1)^k\binom{d-1}{k}\sum_{|\underline{i}|=n-k} u_{\underline{i}}\,.$$

Proof. We claim that $\delta^m S(n,d) = \sum_{k=0}^{m}(-1)^k\binom{m}{k}\sum_{|\underline{i}|=n-k} u_{\underline{i}}$ (where the sum is over $\underline{i} \in \mathbb{N}^d$). For $m=0$ we note $S(n,d) = \sum_{|\underline{i}|=n} u_{\underline{i}}$, and for $m=1$ we have

$$\begin{aligned}\delta S(n,d) &= S(n,d) - S(n-1,d)\\ &= \sum_{|\underline{i}|=n} u(\underline{i}) - \sum_{|\underline{i}|=n-1} u(\underline{i}) = \sum_{k=0}^{1}(-1)^k\binom{1}{k}\sum_{|\underline{i}|=n-k} u_{\underline{i}}\,.\end{aligned}$$

Notice that

$$\delta^m S(n,d) = \delta^{m-1}S(n,d) - \delta^{m-1}S(n-1,d)\,.$$

We will prove the claim via induction, assuming the case $m-1$ holds one has

$$\begin{aligned}\delta^m S(n,d) &= \sum_{k=0}^{m-1}(-1)^k\binom{m-1}{k}\sum_{|\underline{i}|=n-k} u_{\underline{i}} - \sum_{k=0}^{m-1}(-1)^k\binom{m-1}{k}\sum_{|\underline{i}|=n-1-k} u_{\underline{i}}\\ &= \sum_{|\underline{i}|=n} u_{\underline{i}} + \sum_{k=1}^{m-1}(-1)^k\binom{m-1}{k}\sum_{|\underline{i}|=n-k} u_{\underline{i}}\\ &\quad - \sum_{k=0}^{m-2}(-1)^k\binom{m-1}{k}\sum_{|\underline{i}|=n-1-k} u_{\underline{i}} - (-1)^{m-1}\sum_{|\underline{i}|=n-m} u_{\underline{i}}\\ &= \sum_{|\underline{i}|=n} u_{\underline{i}} + \sum_{k=1}^{m-1}(-1)^k\binom{m-1}{k}\sum_{|\underline{i}|=n-k} u_{\underline{i}}\\ &\quad + \sum_{k=1}^{m-1}(-1)^k\binom{m-1}{k-1}\sum_{|\underline{i}|=n-k} u_{\underline{i}} + (-1)^m\sum_{|\underline{i}|=n-m} u_{\underline{i}}\\ &= \sum_{k=0}^{m}(-1)^k\binom{m}{k}\sum_{|\underline{i}|=n-k} u_{\underline{i}}\,.\end{aligned}$$

Thus the claim is true for all m, in particular for $m = d-1$ we have $\delta^{d-1}S(n,d) = u^c_n$. □

Using similar arguments we can show the consistency of the combination technique.

Lemma 2.29 ([110]). *Let*

$$N(n,d) := \sum_{\{\underline{i}\in\mathbb{N}^d : |\underline{i}|=n\}} 1 = \binom{n+d-1}{d-1}$$

then

$$\delta^{d-1}N(n,d) = 1\,.$$

Proof. We claim $\delta^k N(n,d) = N(n,d-k)$. For $k=0$ it is clearly satisfied and for $k=1$ we have

$$\begin{aligned}\delta N(n,d) &= N(n,d) - N(n-1,d)\\ &= \binom{n+d-1}{d-1} - \binom{n+d-2}{d-1} = \binom{n+d-2}{d-2} = N(n,d-2)\,.\end{aligned}$$

Assuming that $\delta^{k-1}N(n,d) = N(n,d-k+1)$ for each n then we have for k by induction

$$\begin{aligned}\delta^k N(n,d) &= \delta^{k-1}N(n,d) - \delta^{k-1}N(n-1,d)\\ &= N(n,d-k+1) - N(n-1,d-k+1)\\ &= \binom{n+d-k}{d-k} - \binom{n+d-k-1}{d-k}\\ &= \binom{n+d-k-1}{d-k-1} = N(n,d-k)\,.\end{aligned}$$

Thus the claim is true for all k and in particular $\delta^{d-1}N(n,d) = N(n,1) = \binom{n}{0} = 1$. □

Note that if $u_{\underline{i}} = 1$ for all $\underline{i}$ then $S(n,d) = N(n,d)$ and therefore As a result of Lemmas 2.28 and 2.29 one has

$$1 = \delta^{d-1}N(n,d) = \sum_{k=0}^{d-1}(-1)^k\binom{d-1}{k}\sum_{|\underline{i}|=n-k} 1\,.$$

We also have a lemma which is the Leibniz rule for our difference operator, a proof is included for completeness.

Lemma 2.30. *Let $f, g : \mathbb{N} \to \mathbb{R}$, then for all $k \leq n$.*

$$\delta^k(f(n)g(n)) = \sum_{j=0}^{k} \binom{k}{j} (\delta^{k-j} f(n-j))(\delta^j g(n)) .$$

Proof. The case $k = 0$ clearly holds and for $k = 1$ we have

$$\begin{aligned} \delta(f(n)g(n)) &= f(n)g(n) - f(n-1)g(n-1) \\ &= f(n)g(n) - f(n-1)g(n) + f(n-1)g(n) - f(n-1)g(n-1) \\ &= (\delta f(n))g(n) + f(n-1)(\delta g(n)) . \end{aligned}$$

Assuming that the assertion is true for $k-1$ then we have

$$\begin{aligned} \delta^k(f(n)g(n)) &= \delta^{k-1}(f(n)g(n)) - \delta^{k-1}(f(n-1)g(n-1)) \\ &= \sum_{j=0}^{k-1} \binom{k-1}{j} \left(\begin{array}{l} (\delta^{k-1-j} f(n-j))(\delta^j g(n)) \\ -(\delta^{k-1-j} f(n-1-j))(\delta^j g(n-1)) \end{array} \right) . \end{aligned}$$

Notice that

$$\begin{aligned} &(\delta^{k-1-j} f(n-j))(\delta^j g(n)) - (\delta^{k-1-j} f(n-1-j))(\delta^j g(n-1)) \\ =&(\delta^{k-1-j} f(n-j))(\delta^j g(n)) - (\delta^{k-1-j} f(n-1-j))(\delta^j g(n)) \\ &+ (\delta^{k-1-j} f(n-1-j))(\delta^j g(n)) - (\delta^{k-1-j} f(n-1-j))(\delta^j g(n-1)) \\ =&(\delta^{k-j} f(n-j))(\delta^j g(n)) + (\delta^{k-1-j} f(n-1-j))(\delta^{j+1} g(n)) , \end{aligned}$$

and further

$$\begin{aligned} &\sum_{j=0}^{k-1} \binom{k-1}{j} (\delta^{k-j} f(n-j))(\delta^j g(n)) \\ &\qquad = (\delta^k f(n))g(n) + \sum_{j=1}^{k-1} \binom{k-1}{j} (\delta^{k-j} f(n-j))(\delta^j g(n)) \end{aligned}$$

and similarly

$$\begin{aligned} &\sum_{j=0}^{k-1} \binom{k-1}{j} (\delta^{k-1-j} f(n-1-j))(\delta^{j+1} g(n)) \\ &\qquad = \sum_{j=1}^{k} \binom{k-1}{j-1} (\delta^{k-j} f(n-j))(\delta^j g(n)) . \end{aligned}$$

Adding the two parts together using the fact that $\binom{k-1}{j} + \binom{k-1}{j-1} = \binom{k}{j}$ for $j = 1, \dots, k-1$ and $\binom{k-1}{j-1} = \binom{k}{j}$ for $j = k$ one has

$$\begin{aligned}\delta(f(n)g(n)) &= (\delta^k f(n))g(n) + \sum_{j=1}^{k} \binom{k}{j} (\delta^{k-j} f(n-j))(\delta^j g(n)) \\ &= \sum_{j=0}^{k} \binom{k}{j} (\delta^{k-j} f(n-j))(\delta^j g(n)) \,,\end{aligned}$$

as required. □

Now we prove a lemma relating to the combination of a quantity whose value depends only upon the level of each component grid.

Lemma 2.31 ([110]). *Led $d \in \mathbb{N}$ and $f : \mathbb{N} \to \mathbb{R}$ and define*

$$F(n) := \sum_{l=0}^{n} \binom{n-l+d-1}{d-1} f_l \,. \tag{2.23}$$

If $0 \leq k < d \leq n$ then $\delta^k F(n) = G_k(n) + H_k(n)$ where

$$\begin{aligned} G_k(n) &:= \begin{cases} 0 & k = 0 \\ \sum_{l=1}^{k} \binom{d-l-1}{k-l} f_{n-k+l} & k \geq 1 \end{cases} \,, \\ H_k(n) &:= \sum_{l=0}^{n-k} \binom{n-k-l+d-1}{d-k-1} f_l \,. \end{aligned}$$

Additionally, $\delta^d F(n) = f_n$.

Proof. For $k = 0$ it is clear that $\delta^0 F(n) = F(n) = H_0(n) = G_0(n) + H_0(n)$. Similarly for $k = 1$ we note that

$$\begin{aligned}\delta F(n) &= F(n) - F(n-1) \\ &= f_n + \sum_{l=0}^{n-1} \left(\binom{n-l+d-1}{d-1} - \binom{n-1-l+d-1}{d-1} \right) f_l \\ &= f_n + \sum_{l=0}^{n-1} \binom{n-l+d-2}{d-2} f_l = G_1(n) + H_1(n) \,.\end{aligned}$$

We will prove the general case by induction. Assume $k > 1$ and $\delta^{k-1} F(n) = G_{k-1}(n) + H_{k-1}(n)$, then

$$\delta^k F(n) = \delta\left(\delta^{k-1} F(n)\right) = \delta(G_{k-1}(n) + H_{k-1}(n)) \,.$$

Looking at $\delta G_{k-1}(n)$ first we have

$$\delta G_{k-1}(n) = \sum_{l=1}^{k-1} \binom{d-l-1}{k-l-1} (f_{n-k+l+1} - f_{n-k+l})$$

and as

$$\begin{aligned}\sum_{l=1}^{k-1} \binom{d-l-1}{k-l-1} f_{n-k+l+1} &= \binom{d-k}{0} f_n + \sum_{l=1}^{k-2} \binom{d-l-1}{k-l-1} f_{n-k+l+1} \\ &= f_n + \sum_{l=2}^{k-1} \binom{d-l}{k-l} f_{n-k+l}\end{aligned}$$

and similarly

$$\sum_{l=1}^{k-1} \binom{d-l-1}{k-l-1} f_{n-k+l} = \binom{d-2}{k-2} f_{n-k+1} + \sum_{l=2}^{k-1} \binom{d-l-1}{k-l-1} f_{n-k+l}$$

then one has

$$\begin{aligned}\delta G_{k-1}(n) &= f_n - \binom{d-2}{k-2} f_{n-k+1} + \sum_{l=2}^{k-1} \binom{d-l-1}{k-l} f_{n-k+l} \\ &= -\binom{d-2}{k-2} f_{n-k+1} - \binom{d-2}{k-1} f_{n-k+1} + \sum_{l=1}^{k} \binom{d-l-1}{k-l} f_{n-k+l} \\ &= -\binom{d-1}{k-1} f_{n-k+1} + G_k(n)\,.\end{aligned}$$

Similarly for $\delta H_{k-1}(n)$ we have

$$\begin{aligned}\delta H_{k-1}(n) &= \sum_{l=0}^{n-k+1} \binom{n-k-l+d}{d-k} f_l - \sum_{l=0}^{n-k} \binom{n-k-l+d-1}{d-k} f_l \\ &= \binom{d-1}{d-k} f_{n-k+1} + \sum_{l=0}^{n-k} \binom{n-k-l+d-1}{d-k-1} f_l \\ &= \binom{d-1}{k-1} f_{n-k+1} + H_k(n)\,.\end{aligned}$$

Thus $\delta^k F(n) = G_k(n) + H_k(n)$ and the hypothesis is true for all $k \geq 0$. For the special case $k = d-1$ we have

$$\delta^{d-1} F(n) = G_{d-1}(n) + H_{d-1}(n) = \left(\sum_{l=1}^{d-1} f_{n-d+1+l}\right) + \left(\sum_{l=0}^{n-d+1} f_l\right) = \sum_{l=0}^{n} f_l$$

and therefore

$$\delta^d F(n) = \sum_{l=0}^{n} f_l - \sum_{l=0}^{n-1} f_l = f_n$$

as required. □

One last Lemma now which relates to the combination of individual error terms in the error splitting.

Lemma 2.32 ([110])**.** *Let $m, d, p \geq 1$, $v : \mathbb{R}_+^m \to \mathbb{R}$ and for $n \in \mathbb{N}$*

$$F(n) := \sum_{\{\underline{i} \in \mathbb{N}^d : |\underline{i}|_1 = n\}} v(2^{-i_1}, \dots, 2^{-i_m}) 2^{-pi_1} \cdots 2^{-pi_m} \,.$$

Then

$$\delta^{d-1} F(n) = 2^{-pn} \sum_{j=0}^{m-1} \binom{m-1}{j} (-2^p)^j s_{n-j}$$

where $s_l := \sum_{\{\underline{i} \in \mathbb{N}^m : |\underline{i}|_1 = l\}} v(2^{-i_1}, \dots, 2^{-i_m})$.

Proof. We begin by noting that we can rewrite $F(n)$ as

$$\begin{aligned} F(n) &= \sum_{\{\underline{i} \in \mathbb{N}^d : |\underline{i}|_1 = n\}} v(2^{-i_1}, \dots, 2^{-i_m}) 2^{-p(i_1 + \cdots + i_m)} \\ &= \sum_{l=0}^{n} \sum_{i_1 + \cdots + i_m = l} v(2^{-i_1}, \dots, 2^{-i_m}) 2^{-pl} \sum_{i_{k+1} + \cdots + i_d = n-l} 1 \\ &= \sum_{l=0}^{n} s_l 2^{-pl} \binom{n-l+d-m-1}{d-m-1} \,. \end{aligned}$$

Noting $F(n)$ is now in the form of (2.23) with $f_l \mapsto s_l 2^{-pl}$ and $d \mapsto d - m$ we have from Lemma 2.31 that

$$\delta^{d-1} F(n) = \delta^{m-1} \left(\delta^{d-m} F(n) \right) = \delta^{m-1} \left(2^{-pn} s_n \right) \,. \tag{2.24}$$

Additionally using Lemma 2.30 we also have

$$\delta^{m-1} (s_n 2^{-pn}) = \sum_{j=0}^{m-1} \binom{m-1}{j} (\delta^{m-1-j} 2^{-p(n-j)}) (\delta^j s_n) \,.$$

Notice that for $k > 0$ one has

$$\delta^k 2^{-pn} = \delta^{k-1} \left(2^{-pn} - 2^{-p(n-1)} \right) = \delta^{k-1} (1 - 2^p) 2^{-pn} = (1 - 2^p) \delta^{k-1} 2^{-pn}$$

from which it follows that $\delta^k 2^{-pn} = (1-2^p)^k 2^{-pn}$. Substituting this back into (2.24) for $k = m-1-j$ one has

$$\begin{aligned}\delta^{m-1}(2^{-pn}s_n) &= \sum_{j=0}^{m-1}\binom{m-1}{j}(1-2^p)^{m-1-j}2^{-p(n-j)}(\delta^j s_n)\\ &= 2^{-p(n-m+1)}\sum_{j=0}^{m-1}\binom{m-1}{j}(2^{-p}-1)^{m-1-j}(\delta^j s_n)\\ &= 2^{-p(n-m+1)}(\delta+2^{-p}-1)^{m-1}s_n\\ &= 2^{-p(n-m+1)}\sum_{j=0}^{m-1}\binom{m-1}{j}(\delta-1)^j 2^{-p(m-1-j)}s_n\,.\end{aligned}$$

Now $(\delta-1)s_n = s_n - s_{n-1} - s_n = -s_{n-1}$ from which it follows that $(\delta-1)^j s_n = (-1)^j s_{n-j}$. Therefore

$$\begin{aligned}\delta^{d-1}F(n) &= 2^{-p(n-m+1)}\sum_{j=0}^{m-1}\binom{m-1}{j}(-1)^j 2^{-p(m-1-j)}s_{n-j}\\ &= 2^{-pn}\sum_{j=0}^{m-1}\binom{m-1}{j}(-2^p)^j s_{n-j}\end{aligned}$$

as required. □

Finally, we have all of the ingredients for the main result.

Proof of Theorem 2.27. From Lemma 2.29 we have that

$$\delta^{d-1}\sum_{|\underline{i}|=n}u = u\delta^{d-1}\sum_{|\underline{i}|=n}1 = u\delta^{d-1}N(n,d) = u\,,$$

and therefore combined with Lemma 2.28 one has

$$u - u_n^c = \delta^{d-1}\sum_{|\underline{i}|=n}u - u_{\underline{i}}\,. \tag{2.25}$$

Letting

$$F_{j_1,\dots,j_m}(n) = \sum_{|\underline{i}|=n}v_{j_1,\dots,j_m}(h_{i_{j_1}},\dots,h_{i_{j_m}})h_{i_{j_1}}^p\cdots h_{i_{j_m}}^p$$

then upon substituting (2.22) into (2.25) and changing the order of summation one has

$$u - u_n^c = \sum_{m=1}^{d}\sum_{\substack{\{j_1,\dots,j_m\}\\ \subset\{1,\dots,d\}}}\delta^{d-1}F_{j_1,\dots,j_m}(n)\,.$$

For each of the $F_{j_1,\dots,j_m}$ we apply Lemma 2.32 resulting in

$$\begin{aligned}
u - u_n^c &= \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} 2^{-pn} \sum_{j=0}^{m-1} \binom{m-1}{j} (-2^p)^j s_{n-j,m} \\
&= 2^{-pn} \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} \sum_{j=0}^{m-1} \binom{m-1}{j} (-2^p)^j s_{n-j,m} \,,
\end{aligned}$$

where $s_{l,m} := \sum_{\substack{\underline{i}\in\mathbb{N}^m \\ |\underline{i}|_1=l}} v_{j_1,\dots,j_m}(h_{i_{j_1}},\dots,h_{i_{j_m}})$. As each $|v_{j_1,\dots,j_m}(h_{i_{j_1}},\dots,h_{i_{j_m}})| \leq K$ then from the triangle inequality

$$|s_{n-j,m}| \leq \binom{n-j+m-1}{m-1} K \quad \Longrightarrow \quad \max_{j=0,\dots,m-1} |s_{n-j,m}| \leq \binom{n+m-1}{m-1} K \,,$$

and it follows that

$$\begin{aligned}
\left| \sum_{j=0}^{m-1} \binom{m-1}{j} (-2^p)^j s_{n-j,m} \right| &\leq K \binom{n+m-1}{m-1} \sum_{j=0}^{m-1} \left| \binom{m-1}{j} (-2^p)^j \right| \\
&\leq K \binom{n+m-1}{m-1} (1+2^p)^{m-1} \,.
\end{aligned}$$

Therefore

$$\begin{aligned}
|u - u_n^c| &\leq 2^{-pn} \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} \left| \sum_{j=0}^{m-1} \binom{m-1}{j} (-2^p)^j s_{n-j,m} \right| \\
&\leq 2^{-pn} \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} K \binom{n+m-1}{m-1} (1+2^p)^{m-1} \\
&= K 2^{-pn} \sum_{m=1}^{d} \binom{d}{m} \binom{n+m-1}{m-1} (1+2^p)^{m-1} \qquad (2.26) \\
&\leq K 2^{-pn} (1+2^p)^{d-1} \sum_{m=1}^{d} \binom{d}{m} \binom{n+m-1}{m-1} \,.
\end{aligned}$$

Now given the Vandermonde type identity

$$\begin{aligned}\sum_{k=0}^{n+2d-1}\binom{n+2d-1}{k}x^k &= (1+x)^{n+2d-1}\\ &= (1+x)^d(1+x)^{n+d-1}\\ &= \left(\sum_{i=0}^{d}\binom{d}{i}x^i\right)\left(\sum_{j=0}^{n+d-1}\binom{n+d-1}{j}x^j\right)\\ &= \sum_{k=0}^{n+2d-1}x^k\left(\sum_{m=0}^{k}\binom{d}{m}\binom{n+d-1}{k-m}\right),\end{aligned}$$

consider the x^{d-1} term, one has

$$\binom{n+2d-1}{d-1}=\sum_{m=0}^{d-1}\binom{d}{m}\binom{n+d-1}{d-1-m}=\sum_{m=1}^{d}\binom{d}{d-m}\binom{n+d-1}{m-1}$$

where the second equality is obtained by the substitution $m \mapsto m-d$. Since $\binom{d}{d-m}=\binom{d}{m}$ then

$$|u-u_n^c| \leq K2^{-pn}(1+2^p)^{d-1}\binom{n+2d-1}{d-1},$$

as required. □

Note that for $d=1$ and $p=2$ our bound reduces to $|u-u_n^c| \leq K2^{-2n}$ as one would expect from a tight bound. For $d=2$ and $p=2$ we obtain $|u-u_n^c| \leq K2^{-2n}5(n+3)$ compared to the $(3+5n)Kh_n^2$ derived in Lemma 2.25. Similarly for $d=3$ and $p=2$ Theorem 2.27 gives the bound $|u-u_n^c| \leq K2^{-2n}\frac{25}{2}(n+5)(n+4)$ compared to the $\left(7+\frac{25}{2}n+\frac{25}{2}n^2\right)Kh_n^2$ of Lemma 2.26. In both cases we see the leading error term is consistent with the result of the previous lemmas whilst the non-leading error terms are larger. Thus Theorem 2.27 sacrifices some tightness for generality but still provides an accurate asymptotic result. For a given d one may expand (2.26) to obtain better estimates of the non-leading error terms.

2.2.4 Other work and additional remarks

There have been many other error analyses of the combination technique in the literature, see for example [104, 58] which consider the combination of Ritz-Galerkin and Galerkin projections respectively. We have focused on error analysis based on error splittings because of our interest in finite difference methods for hyperbolic

PDEs. It has been shown that the error splitting model is satisfied for the numerical solutions of the Laplace equation [23, 24, 25] and also for a first order implicit advection scheme in which time was treated as another spatial variable [110]. In Chapter 3 we will discuss whether more general hyperbolic problems satisfy the error splitting model and also review an analysis of leading order error terms as in [82].

In Lemmas 2.25, 2.26 and Theorem 2.27 it was assumed that the order of convergence was the same for all of the spatial variables and each of the terms in the error splitting were bounded by the same constant. In practice there are applications for which the order of convergence may be different and/or the constant terms may differ significantly in magnitude. For example consider physical problems defined in phase space for which it is not unreasonable to expect that convergence with respect to the velocity dimensions is different from that in the spatial dimensions. This was considered for 2 dimensional Galerkin projections in [58] with a modified combination technique that allows one to skew the layers in order to compensate for different rates of convergence.

The combination technique presented thus far is a very specific combination of component solutions for a given dimension and level. In practice however, there are several variations of the combination technique which are used. One such variation is the truncated combination [9]. For some problems the strongly anisotropic grids, that is those with coarse grid spacing in one or more dimensions and much finer spacing in the remaining dimensions, may resolve the solution poorly resulting in inaccurate extrapolation when the combination technique is applied. To avoid this problem the truncated combination technique excludes these strongly anisotropic grids from the sum. Another variation is combinations onto dimension adaptive sparse grids. Here we choose our combination grids and coefficients differently such that unknowns are only used where they provide the most benefit. Several other variations and generalisations exist. We will discuss some of these further in Chapter 4.

The combination technique has several computational advantages over direct sparse grid approaches. First, there is no need to compute using hierarchical bases. The approximation on each component/coarse grid can be computed in almost any way, e.g. finite difference methods, finite element methods, finite volume methods, to name a few, as long as the same method is used for all of the component/coarse grids. In particular this means that no modifications are required for existing grid based solvers if one wanted to try the combination technique. Second, each component/coarse approximation can be computed independently. This makes the combination technique embarrassingly parallel with respect to the computation of different $u_{\underline{i}}$. As a result, the combination technique has the potential to be easily adapted to existing applications whilst also providing additional parallelism.

Chapter 3

Hyperbolic PDEs

Much of the classical combination technique literature has focused on solving elliptic PDEs. Whilst many recent papers have published numerical results for the combination technique applied to a wider range of problems, there has been limited analysis of the combination technique for these problems. Many of the largest problems being solved on HPCs are physical problems which evolve over time. These are typically modelled by hyperbolic or parabolic PDEs. Our focus is on the former of the two but it should be noted that much of the work presented here applies equally well to parabolic problems. In Section 3.1 we give a brief background on hyperbolic PDEs. Whilst there are numerous examples of hyperbolic PDEs our attention is focused on the advection PDE which is used in subsequent chapters for testing generalisations of the combination technique (Chapter 4) and fault tolerant PDE solvers based on the combination technique (Chapter 5). Following this, Section 3.2 reviews finite difference schemes used to solve the advection equation covering important properties like order of approximation and stability. Of particular interest are errors which are dependent on the time stepping as these are not present in the analysis of the combination technique in most of the literature. We give some proofs of convergence and stability of some classical schemes for illustrative purposes. Whilst the techniques are relatively standard and well-known they are presented here with a little more care than most texts on the subject. In Section 3.3 we look at the application of the combination technique to the solution of time-dependent PDEs particularly focusing on numerical aspects like stability, order of approximation and error expansions. We provide results which extend and make more precise the work found in [82] which roughly shows finite difference solutions of the advection equation satisfy an error splitting. Whilst the techniques are relatively standard the results are original to the extent described.

3.1 Background on Hyperbolic PDEs

The wave equation,

$$\frac{\partial^2}{\partial t^2}u - c^2\frac{\partial^2}{\partial x^2}u = 0$$

where $c \in \mathbb{R}$ and $u = u(x,t) : \Omega \times \mathbb{R}_+ \to \mathbb{R}$ with $\Omega \subset \mathbb{R}$ and $\mathbb{R}_+ := [0,\infty)$, is typically given as the prototypical hyperbolic PDE. For simplicity we assume functions are real valued and consider initial value problems moving forward in time from an initial condition given at time $t = 0$. For finite domains we typically consider periodic boundary conditions. The wave equation is extremely important to science, particularly physics, as it models many physical waves of interest including pressure waves (e.g. sound in air) and electromagnetic waves (via Maxwell's equations). The wave equation may be factored into the form

$$\left(\frac{\partial}{\partial t} + c\frac{\partial}{\partial x}\right)\left(\frac{\partial}{\partial t} - c\frac{\partial}{\partial x}\right)u = 0\,.$$

Now transforming to characteristic coordinates $u(x,t) \mapsto u(\xi,\eta)$ where $\xi = x - ct$ and $\eta = x + ct$ one obtains

$$\frac{\partial^2}{\partial \xi \partial \eta}u = 0\,.$$

Solutions to this equation clearly have the form

$$u(x,t) = F(\xi) + G(\eta) = F(x-ct) + G(x+ct)$$

where $F, G : \mathbb{R} \to \mathbb{R}$. Notice that $F(x-ct)$ is a solution to the PDE

$$\left(\frac{\partial}{\partial t} + c\frac{\partial}{\partial x}\right)F(x-ct) = -cF'(x-ct) + cF'(x-ct) = 0 \tag{3.1}$$

(where $F'(\xi) := \frac{d}{d\xi}F(\xi)$). The PDE (3.1) is the advection equation in one spatial dimension. The advection equation is a special case of more general conservation equations which may be written as

$$\frac{\partial}{\partial t}u + \frac{\partial}{\partial x}f(u,x,t) = 0\,, \tag{3.2}$$

where $f(u,x,t)$ is sometimes referred to as the flux of u.

The (inviscid) Burgers' equation is one of the simplest examples of a non-linear hyperbolic PDE. It is obtained from the conservation equation (3.2) with $f(u,x,t) = \frac{1}{2}u^2$ for which one may write

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) . \tag{3.3}$$

Solutions may again be obtained via the method of characteristics, namely given an initial condition $u(x,0) = u_0(x)$, then a given point x at $t = 0$ travels at constant speed $u_0(x)$. It follows that $u(x,t) = u_0(x - u(x,t)t)$. However, there is an implicit assumption made here that the characteristics are well behaved, in particular that they do not intersect or diverge. This is not always the case and rarefactions and shocks can occur depending upon $u_0(x)$. We give two examples.

Example 3.1. Consider the solution of (3.3) on the spatial domain $\Omega = \mathbb{R}$ for $t \geq 0$ with the initial condition

$$u_0(x) = \begin{cases} 1 & x \leq 0 \\ 1 - x & 0 < x < 1 \\ 0 & x \geq 1 \, . \end{cases}$$

One sees that the function values for $x \leq 0$ propagate forwards at a speed of 1, for $x \geq 1$ do not move, and for $0 < x < 1$ move forward at speed $1 - x$ such that as $t \to 1$ they all converge towards the location $x = 1$. In particular for $0 \leq t < 1$ one has

$$u(x,t) = \begin{cases} 1 & x \leq t \\ 1 - \frac{x-t}{1-t} & t < x < 1 \\ 0 & x \geq 1 \, , \end{cases}$$

which converges to the discontinuous solution (shock) $u(x,1) = \chi_{(-\infty,1]}(x)$. For $t \geq 1$ a classical solution does not exist but in practice one might continue to propagate the non-zero portion of the solutions such that $u(x,t) = \chi_{(-\infty,t]}(x)$ for $t \geq 1$. It is straightforward to show this solution satisfies (3.3) for $t \geq 1$ given the initial condition $u(x,1) = \chi_{(-\infty,1]}(x)$.

Example 3.2. Consider the solution of (3.3) on the spatial domain $\Omega = \mathbb{R}$ for $t \geq 0$ with the initial condition $u_0(x) = \chi_{[0,\infty)}(x)$. Here we see that for $x < 0$ the solution is zero and does not propagate but for $x \geq 1$ the solution propagates forwards with a speed of 1 such that $u(x) = 0$ for $x < 0$ and $u(x) = 1$ for $x \geq t$. Here the characteristics diverge such that there is no classical solution

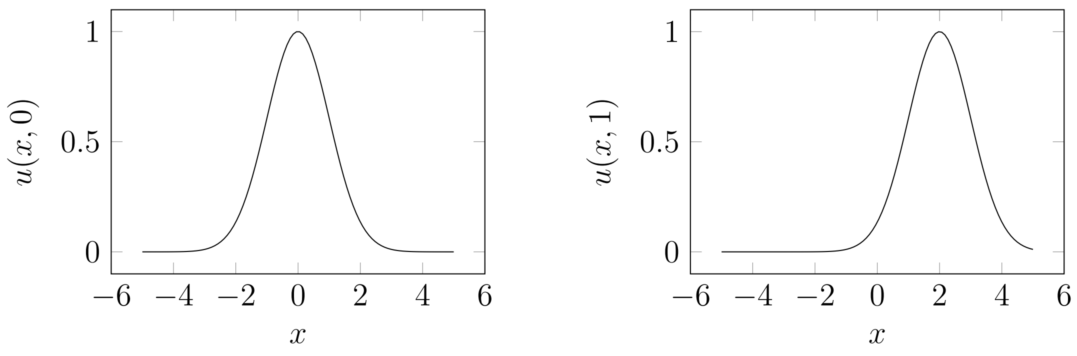


Figure 3.1: *Here we depict a solution to the scalar advection equation* $\frac{\partial u}{\partial t} + 2\frac{\partial u}{\partial x} = 0$ *on the domain* $\mathbb{R}$ *with initial condition* $u(x,0) = e^{-x^2/2}$ *which is depicted on the left. The solution at time* $t = 1$ *is depicted on the right and is simply a translation of the Gaussian function* 2 *units to the right.*

for $0 \leq x < t$. This is a case of a rarefaction and in practice one may linearly interpolates in this interval, that is

$$u(x,t) = \begin{cases} 0 & x \leq 0 \\ x/t & 0 < x < t \\ 1 & x \geq t\,, \end{cases}$$

for which it is easily checked that this satisfies Burgers' equation (in the interval $(0,t)$ one has $\frac{\partial u}{\partial t} = \frac{-x}{t^2}$ and $u\frac{\partial u}{\partial x} = \frac{x}{t}\frac{1}{t}$).

These examples demonstrate some features that sometimes occur in hyperbolic PDEs that make hyperbolic PDE solvers more difficult to develop compared to parabolic or elliptic PDEs where these features do not occur. The combination technique has been applied to hyperbolic PDEs for which shocks occur [81]. The ability to resolve these typically appears to depend on the location and direction of the shock due to the sparse nature of local regions of a classical sparse grid. It is an open area to study the combination technique in more detail for such problems. Whilst being an interesting area of study this is not addressed in this thesis and as such we focus on problems for which shocks and rarefactions do not occur. In particular, we focus on problems for which we can write $f(u,x,t) = a(x,t)u$ and $a(x,t) \in C^1(\Omega \times \mathbb{R}_+)$. In Figure 3.1 we depict an example on the domain $\Omega = \mathbb{R}$ with constant $a = 2$ and initial condition $u(x,0) = e^{-x^2/2}$.

We consider the solution to the advection equation over many spatial dimensions. Let $0 < d \in \mathbb{N}$ be the number of spatial dimensions, $\Omega \subset \mathbb{R}^d$ be the domain and $u = u(\boldsymbol{x},t) : \Omega \times [0,\infty) \mapsto \mathbb{R}$. Given $\boldsymbol{a} \in \mathbb{R}^d$ one has the (scalar) advection equation

$$\frac{\partial u}{\partial t} + \nabla \cdot (\boldsymbol{a} u) = 0 \,. \tag{3.4}$$

For constant $\boldsymbol{a}$ and initial condition $u(\boldsymbol{x}, 0) := u_0(\boldsymbol{x})$ one has $\nabla \cdot (\boldsymbol{a}u) = \boldsymbol{a} \cdot \nabla u$ and the solution $u(\boldsymbol{x}, t) = u_0(\boldsymbol{x} - \boldsymbol{a}t)$ is easily obtained via the method of characteristics. More generally $\boldsymbol{a}$ could be a (vector valued) function of $\boldsymbol{x}, \boldsymbol{t}$. A noteworthy example is a stationary velocity vector field $\boldsymbol{a} = \boldsymbol{a}(\boldsymbol{x})$ for which $\nabla \boldsymbol{a}(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \Omega$. This corresponds to an incompressible flow and such $\boldsymbol{a}(\boldsymbol{x})$ is often called a solenoidal vector field. Again one has $\nabla \cdot (\boldsymbol{a}u) = \boldsymbol{a} \cdot \nabla u$ and the method of characteristics may be used to obtain a solution, that is by solving the ordinary differential equation

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{a}(\boldsymbol{x}(t)) \quad \Longrightarrow \quad \boldsymbol{x}(t) = \boldsymbol{x}(0) + \int_0^t \boldsymbol{a}(\boldsymbol{x}(t))\, dt \,,$$

and setting $u(\boldsymbol{x}(t), t) = u(\boldsymbol{x}(0), 0)$.

Adding a diffusion term to (3.4) one obtains the advection-diffusion equation

$$\frac{\partial u}{\partial t} + \nabla \cdot (\boldsymbol{a} u) - \nabla \cdot (D \nabla u) = 0 \,, \tag{3.5}$$

where $D > 0$ is the diffusivity. This may be used to describe the diffusion of ink in water flowing down a channel. We do not study this in any detail but test some of our fault tolerant algorithms on this PDE as well. The advection diffusion PDE is often easier to solve numerically than a pure advection equation. The reason for this is that the diffusion term leads to greater stability in numerical schemes. For example, most explicit advection solvers require some artificial diffusion to achieve stability whilst for an advection-diffusion solver there is less need for artificial diffusion because of the presence of diffusion term in the actual problem.

We now briefly describe some hyperbolic problems that are fundamental to the physical sciences. Maxwell's equations for a vacuum (no source or current) are an example of a wave equation and are given by

$$\begin{aligned} \nabla \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \,, \qquad & \nabla \times \boldsymbol{B} = \frac{1}{c^2}\frac{\partial \boldsymbol{E}}{\partial t} \,, \\ \nabla \cdot \boldsymbol{E} = 0 \,, \qquad & \nabla \cdot \boldsymbol{B} = 0 \,, \end{aligned}$$

with c being the speed of light, and $\boldsymbol{E}$, $\boldsymbol{B}$ being the electric and magnetic fields respectively. Taking the curl of the first equation and substituting the second yields

$$\nabla \times (\nabla \times \boldsymbol{E}) = -\frac{\partial \nabla \times \boldsymbol{B}}{\partial t} = -\frac{1}{c^2}\frac{\partial^2 \boldsymbol{E}}{\partial t^2} \,.$$

As

$$\nabla \times (\nabla \times \boldsymbol{E}) = \nabla(\nabla \cdot \boldsymbol{E}) - \nabla^2 \boldsymbol{E} = -\nabla^2 \boldsymbol{E}$$

(with $\nabla^2 \boldsymbol{E} = (\nabla^2 E_1, \dots, \nabla^2 E_d)$) and observing one can do the same for $\boldsymbol{B}$ then one reduces Maxwell's equations in a vacuum to the wave equation in each component.

Boltzmann equations are used to describe transport in thermodynamic systems. A relatively general form is given by

$$\frac{\partial u}{\partial t} + \frac{\boldsymbol{p}}{m} \cdot \nabla_{\boldsymbol{x}} u + \boldsymbol{F} \cdot \nabla_{\boldsymbol{p}} u = \left(\frac{\partial u}{\partial t}\right)_{\text{coll}}, \tag{3.6}$$

where $\nabla_{\boldsymbol{x}} u := \left(\frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_d}\right)$ and similarly for $\nabla_{\boldsymbol{p}} u$. Here $\boldsymbol{p}$ is the momentum coordinates, m is the mass, $\boldsymbol{F}$ is the force acting on the particles and u is a (probability density) function over $\boldsymbol{x}, \boldsymbol{p}, t$. Without the collision term $\left(\frac{\partial u}{\partial t}\right)_{\text{coll}}$ this is often referred to as the Vlasov equations. The Vlasov equations are effectively an advection equation in 6 dimensional phase space with the velocity field given by $\frac{\boldsymbol{p}}{m}$ and $\boldsymbol{F}$ for the spatial and momentum coordinates respectively. For the Vlasov-Maxwell system the force $\boldsymbol{F}$ is determined by $q(\boldsymbol{E} + \frac{1}{mc}(\boldsymbol{p} \times \boldsymbol{B}))$ where c is the speed of light, q is the electric charge of each particle, and $\boldsymbol{E}$ and $\boldsymbol{B}$ are the electric and magnetic fields obtained by solving the Maxwell equations over the charged field. When there are many species (i.e. types of particles) then we have a system of PDEs, i.e. one PDE for each species, which interact via the electric and magnetic fields (and collisions if included). The GENE code (`genecode.org`) solves a related system of equations for simulating plasma turbulence. A gyrokinetic transformation is applied to the Vlasov-Maxwell system which averages over the gyromotion (fast cyclic motion) of the particles to obtain a 6 dimensional system of equations (5 in space compared to the 6 for the Vlasov-Maxwell system). In the papers [5, 72] several numerical experiments using the large scale and complex GENE code have been presented. These results validate the fault tolerant adaptation of the combination technique developed later in this thesis. However, the study of these systems of equations is not the focus of this thesis and we therefore refrain from discussing them in depth. For a detailed discussion of the GENE code we refer the reader to [54]. In this thesis we assume that our hyperbolic problems have solutions which are suffiently smooth and that the velocity (and forces) are locally constant such that they may be treated like advection.

3.2 Finite Difference Schemes for Solving Hyperbolic PDEs

There are several classes of numerical methods for solving PDEs which include finite difference, finite volume, finite element and meshless methods. Finite difference is perhaps the simplest to implement, typically using Taylor series expansions around each vertex of a regular grid to obtain a discrete approximation to the PDE. Finite volume methods can also be derived using Taylor series expansions, but applied to the average density over each cell in a mesh. Function densities are transported over cell boundaries and by equating the outgoing flux of one cell with the incoming flux of the corresponding neighbours one develops conservative methods. Finite element methods are designed to find an approximation to the PDE on a given finite dimensional function space, for example the space of piecewise linear functions on a given mesh (typically of simplices). By representing the discrete function space by nodal basis functions and substituting into the weak formulation of the PDE one derives a system of equations which one then solves to obtain an approximate solution via the Galerkin method. Meshless methods is another wide class of methods which do not require a grid or mesh of any kind (as the name suggests).

As the combination technique combines approximate solutions from nested regular grids we focus on solving the advection equation on such grids for which finite difference methods are a natural choice. Whilst finite volume and finite element methods may also be applied to the advection equation on nested regular grids we do not consider these in this thesis. In this section we have two aims. First we discuss the classical approach to the analysis of finite difference schemes for hyperbolic PDEs. In particular we point out why classical convergence results are not enough to guarantee convergence of the combination technique. Second, we examine some classical schemes for the scalar advection equation in one spatial dimension and demonstrate how these may be extended to the solution of the scalar advection equation in higher dimensions. This is a challenge as a naive extension of many one dimensional schemes to higher dimensions leads to unconditionally unstable schemes.

3.2.1 Classical convergence analysis

We define an initial value problem as follows.

Definition 3.3. Consider a PDE of the form

$$\frac{\partial u}{\partial t} = Au \tag{3.7}$$

where A is a linear operator which transforms the (possibly vector) function $u(\boldsymbol{x}, t) : \mathbb{R}^d \times \mathbb{R}_+$ to some other function $Au(\boldsymbol{x}, t) : \mathbb{R}^d \times \mathbb{R}_+$ via spatial differentiations, matrix-vector multiplications and the like. The initial value problem is to find a one-parameter set of functions $u(t) := u(\boldsymbol{x}, t)$ which satisfy (3.7) for $0 \le t \le T$ given the initial state $u(0) = u_0$.

For simplicity we assume that A does not depend on t. Further we do not discuss boundary conditions in this thesis and for problems on bounded domains it will generally be assumed that solutions are periodic. Before we can discuss solutions of such problems it is important that they are well-posed.

Definition 3.4. An initial value problem is well-posed if a solution exists, is unique and depends continuously on the initial condition.

Remark 3.5. In this brief discussion we will not specify the function spaces involved. I feel these details are poorly covered by the modern literature but at the same time a detailed description of the theory in terms of Banach spaces is beyond the scope of this discussion. For a thorough treatment of the subject we refer the reader to the early literature, particularly [84, 112]. Typically the function space will be one which permits a Fourier analysis, for example a Hilbert space.

Classical convergence analysis for hyperbolic PDEs is based on the notions of consistency and stability. Lax and Richtmyer consider the discretisation of (3.7) to the finite difference equations

$$u(\boldsymbol{x}, t + \Delta t) \approx B(\Delta t, \Delta x_1, \ldots, \Delta x_d) u(\boldsymbol{x}, t)$$

where B is a linear finite difference operator, that is a superposition of shift operators which shift by multiples of Δx_k in the x_k direction for each $k = 1, \ldots, d$ with coefficients depending on $\Delta t, \Delta x_1, \ldots, \Delta x_d$, which is bounded for any fixed $\Delta t, \Delta x_1, \ldots, \Delta x_d$. It is assumed that $\Delta x_k = g_k(\Delta t)$ with $g_k(\Delta t) \to 0$ as $\Delta t \to 0$ for $k = 1, \ldots, d$ and we define

$$C(\Delta t) := B(\Delta t, g_1(\Delta t), \ldots, g_d(\Delta t)) .$$

Here we provide an abbreviated definition of consistency, stability and convergence as defined by Lax and Richtmyer in [84].

Definition 3.6. Consider a well-posed initial value problem and a finite difference approximation $C(\Delta t)$.

- $C(\Delta t)$ is a 'consistent' approximation to the solution of the initial value problem if
$$\lim_{\Delta t \to 0} \left\| \frac{C(\Delta t)u(t) - u(t)}{\Delta t} - Au(t) \right\| = 0 \quad \text{uniformly in } t \text{ for } 0 \leq t \leq T$$
(where the norm is such that the function space of solutions is complete, typically L^2, see Remark 3.5).

- $C(\Delta t)$ is 'stable' if for some $\tau > 0$ the set of operators
$$\{C(\Delta t)^n\}_{0<\Delta t \leq \tau,\, 0 \leq n\Delta t \leq T}$$
is uniformly bounded.

- $C(\Delta t)$ is a 'convergent' approximation for the initial value problem if for any sufficiently smooth u_0 and any sequences $\Delta t_j, n_j$ such that $\Delta t_j \to 0$ and $n_j \Delta t_j \to t$ where $0 \leq t \leq T$ then
$$\|C(\Delta t_j)^{n_j} u_0 - u(t)\| \to 0$$
where here $u(t)$ denotes the exact solution to the initial value problem at time t (and again the norm completes the underlying function space).

When the solution space permits a Fourier analysis, e.g. L^2, then stability may be shown via a von Neumann stability analysis [123]. This involves showing that no Fourier mode grows over time in the finite difference equations, or equivalently that the eigenvalues of $C(\Delta t)$ are no more than 1 (or even $1 + \mathcal{O}(\Delta t)$). Stability is typically only satisfies for Δt sufficiently small depending on the width spatial discretisation $\Delta x_1, \ldots, \Delta x_d$. This was first observed by Courant, Friedrichs and Lewy [34] and is often referred to as the CFL condition. We may now give the Lax–Richtmyer equivalence theorem.

Theorem 3.7 ([84, 112])**.** *Given a well-posed initial value problem and consistent finite difference approximation $C(\Delta t)$ then stability is a necessary and sufficient condition for $C(\Delta t)$ to be convergent.*

Whilst the above equivalence theorem is very nice and has formed the foundations of numerical analysis of hyperbolic PDEs it does not guarantee that the combination technique will converge when applied to a convergent scheme. To illustrate this let $B(\Delta t, \Delta x_1, \dots, \Delta x_d)$ be a finite difference approximation of A such that the corresponding $C(\Delta t)$ is convergent. Fix $0 \leq t \leq T$ and a positive integer m. Now let $\Delta t = t/m$ and $u_{\underline{i}}(t) = B(\Delta t, 2^{-i_1}, \dots, 2^{-i_d})^m u_0$. As $C(\Delta t)$ is convergent then the approximation $u_{\underline{i}}(t)$ of $u(t)$ converges as $m, i_1, \dots, i_d \to \infty$. Recall the combination technique of level n is given by

$$u_n^c := \sum_{k=0}^{d-1} (-1)^d \binom{d-1}{k} \sum_{|\underline{i}|=n-k} u_{\underline{i}} \,.$$

Now it is clear that if *all* of the $u_{\underline{i}}$ converged as $n \to \infty$ then u_n^c also converges (as the sum of combination coefficients is 1). Unfortunately, all of the $u_{\underline{i}}$ do not converge. For example, consider $\underline{i} = (n, 0, \dots, 0)$. Clearly as $n \to \infty$ the corresponding $u_{\underline{i}}$ does not necessarily converge to $u(t)$. Thus for convergence of the classical combination technique it is not enough that our numerical schemes are convergent. In particular, one typically requires that the error of our approximation satisfies an error splitting formula as in Theorem 2.27. These error splittings must be satisfied pointwise and therefore we effectively require that our numerical schemes converge pointwise. This is much stronger than the usual notions of convergence and as such we typically require additional smoothness of our initial condition and solution. In Section 3.3 we show that a particular class of finite difference approximations for the scalar advection equation do indeed satisfy this error splitting.

Remark 3.8. We could consider the starting combination u_n^c with which we then refine all of the components $u_{\underline{i}}$, that is by incrementing each i_k by 1 at each refinement. In this scenario convergence of the combination does indeed follow as as the number of refinements $n' \to \infty$ then each $u_{\underline{i}+\underline{n}'}$ converges. However, these combinations correspond to a sequence of truncated combinations which will be discussed separately in Section 4.1.

3.2.2 Numerical schemes for advection

Here we consider the PDE (3.2) in one spatial dimension and the PDE (3.4) in $d > 1$ spatial dimensions. For any analysis we will typically assume that $f(u) = au$ with constant $a \in \mathbb{R}$ and constant $\boldsymbol{a} \in \mathbb{R}^d$ for each of the PDEs respectively. As in Section 2.2 we consider the domain $[0, 1]^d$ discretised into level $\underline{l}$ grids $\Omega_{\underline{l}}$ with vertices $(j_1 2^{-l_1}, \dots, j_d 2^{-l_d})$ with $\underline{0} \leq \underline{j} \leq 2^{\underline{l}}$. For the finite difference methods that follow we consider discrete times $t_n = n\Delta t$ and grid points $x_j = j\Delta x$ for problems in one spatial dimension and $x_{\underline{j}} = (j_1\Delta x_1, \dots, j_d\Delta x_d)$ more generally. We use the notation $U^n_{\underline{j}}$ to denote the finite difference approximation of the function u at the point $x_{\underline{j}}$ at time t_n, that is $U^n_{\underline{j}} \approx u(x_{\underline{j}}, t_n)$. We emphasise here the difference between $U^n_{\underline{j}}$, the finite difference approximation of u and some point on a grid, and $u_{\underline{j}}$, the approximation of u discussed in the context of the combination technique. Similarly F^n_j denotes an approximation to the flux $f(u(x_j,t_n), x_j, t_n)$. The difference between the approximation and the exact value of u is denoted by $\epsilon^n_{\underline{j}} := U^n_{\underline{j}} - u(x_{\underline{j}}, t_n)$.

First order schemes

We first look at first order schemes for the advection equation. The upwind, or forward time backward space (FTBS), discretisation of the scalar advection PDE in one spatial dimension (3.2) is given by

$$\frac{U^{n+1}_j - U^n_j}{\Delta t} + \frac{F^n_j - F^n_{j-1}}{\Delta x} = 0\,, \tag{3.8}$$

which is used when $\frac{\partial f}{\partial u} > 0$, that is transport is in the positive x direction. For transport in the opposite direction one replaces $F^n_j - F^n_{j-1}$ with $F^n_{j+1} - F^n_j$ to obtain a forward time forward space (FTFS) scheme. Re-arranging (3.8) yields the explicit update formula

$$U^{n+1}_j = U^n_j - \frac{a\Delta t}{\Delta x}(F^n_j - F^n_{j-1})\,.$$

If $f(u,x,t) = au$ with constant $a > 0$ then a von Neumann stability analysis shows that this method is stable iff $0 \leq \frac{a\Delta t}{\Delta x} \leq 1$. The Godunov scheme is a closely related discretisation for finite volume methods where one solves the Riemann problem over cell boundaries to obtain the flux terms. With $\frac{\partial f}{\partial u} > 0$ the Godunov scheme reduces to the upwind scheme given. The following proposition tells us that the FTBS scheme is in fact pointwise convergent given a sufficiently smooth initial condition and constant velocity.

Proposition 3.9. *The numerical scheme* (3.8) *with* $F^n_j = aU^n_j$ *and constant* $a \in \mathbb{R}_+$ *for solving the* PDE (3.2) *with* $f(u) = au$ *and initial condition* $u(x,0) \in W^{2,\infty}(\mathbb{R})$ *is pointwise convergent if* $0 < a\Delta t \leq \Delta x$ *and* $U^0_j = u(x_j, 0)$. *Further, with* $t_n = n\Delta t$ *and* $x_j = j\Delta x$ *one has the error bound*

$$\left|u(x_j,t_n) - U^n_j\right| \leq t_n \frac{a\Delta x + a^2\Delta t}{2} \left\|\frac{\partial^2 u(x,0)}{\partial x^2}\right\|_\infty .$$

Proof. Using a Taylor series expansion over time around t_n one obtains

$$\begin{aligned}
\epsilon_j^{n+1} &:= U_j^{n+1} - u(x_j, t_{n+1}) \\
&= U_j^n - \frac{a\Delta t}{\Delta x}(U_j^n - U_{j-1}^n) - \left(u(x_j,t_n) + \Delta t \frac{\partial}{\partial t}u(x_j,t_n) + \frac{\Delta t^2}{2}\frac{\partial^2}{\partial t^2}u(x_j,\xi_t)\right) \\
&= \epsilon_j^n - \frac{a\Delta t}{\Delta x}\left(U_j^n - U_{j-1}^n - \Delta x \frac{\partial}{\partial x}u(x_j,t_n)\right) - \frac{\Delta t^2}{2}\frac{\partial^2}{\partial t^2}u(x_j,\xi_t)\,,
\end{aligned}$$

for some $\xi_t \in [t_n, t_{n+1}]$. Now using a Taylor expansion over x around x_j we obtain

$$\begin{aligned}
\epsilon_j^{n+1} &= \epsilon_j^n - \frac{a\Delta t}{\Delta x}\left(\epsilon_j^n - \epsilon_{j-1}^n - \frac{\Delta x^2}{2}\frac{\partial^2}{\partial x^2}u(\xi_x,t_n)\right) - \frac{\Delta t^2}{2}\frac{\partial^2}{\partial t^2}u(x_j,\xi_t) \\
&= \left(1 - \frac{a\Delta t}{\Delta x}\right)\epsilon_j^n + \frac{a\Delta t}{\Delta x}\epsilon_{j-1}^n + \frac{a\Delta x\Delta t}{2}\frac{\partial^2}{\partial x^2}u(\xi_x,t_n) - \frac{\Delta t^2}{2}\frac{\partial^2}{\partial t^2}u(x_j,\xi_t)\,,
\end{aligned}$$

for some $\xi_x \in [x_{j-1}, x_j]$. Notice that if $0 < a\Delta t \le \Delta x$ then $\left(1 - \frac{a\Delta t}{\Delta x}\right)\epsilon_j^n + \frac{a\Delta t}{\Delta x}\epsilon_{j-1}^n$ is a convex combination. It follows that

$$\left|\left(1 - \frac{a\Delta t}{\Delta x}\right)\epsilon_j^n + \frac{a\Delta t}{\Delta x}\epsilon_{j-1}^n\right| \le \max\{|\epsilon_j^n|, |\epsilon_{j-1}^n|\} \le \max_j |\epsilon_j^n|\,,$$

and therefore

$$|\epsilon_j^n| \le \max_j |\epsilon_j^{n-1}| + \frac{a\Delta x\Delta t}{2}\left|\frac{\partial^2}{\partial x^2}u(\xi_x,t_n)\right| + \frac{\Delta t^2}{2}\left|\frac{\partial^2}{\partial t^2}u(x_j,\xi_t)\right|\,.$$

Now as $\frac{\partial u}{\partial t} = -a\frac{\partial u}{\partial x}$ one has $\frac{\partial^2 u}{\partial t^2} = a^2\frac{\partial^2 u}{\partial x^2}$. Additionally, as $u(x,t) = u(x - at, 0)$ for all x, t one has $\left\|\frac{\partial^2 u(\cdot,t)}{\partial x^2}\right\|_\infty = \left\|\frac{\partial^2 u(\cdot,0)}{\partial x^2}\right\|_\infty$ for all $t \ge 0$. Thus

$$|\epsilon_j^n| \le \max_j |\epsilon_j^{n-1}| + \frac{a\Delta x\Delta t + a^2\Delta t^2}{2}\left\|\frac{\partial^2}{\partial x^2}u(x,0)\right\|_\infty\,,$$

and substituting the result into the right hand side recursively one obtains

$$|\epsilon_j^n| \le \max_j |\epsilon_j^0| + n\frac{a\Delta x\Delta t + a^2\Delta t^2}{2}\left\|\frac{\partial^2}{\partial x^2}u(x,0)\right\|_\infty\,.$$

Note that $t_n = n\Delta t$ and therefore given the initial values are exact ($\epsilon_j^0 = 0$ for all j) then

$$|\epsilon_j^n| \le t_n\frac{a\Delta x + a^2\Delta t}{2}\left\|\frac{\partial^2}{\partial x^2}u(x,0)\right\|_\infty$$

from which it is clear that the method is first order pointwise convergent. □

Notice that the error grows linearly with respect to t. Also notice that the scheme will still be first order pointwise convergent if we relax the initial values to satisfy $U_j^0 = u(x_j, 0) + \mathcal{O}(\Delta x)$.

The FTBS scheme has a natural extension for solving the advection equation in higher dimensions, given the constant velocity vector $\boldsymbol{a} = (a_1, \ldots, a_d) \in \mathbb{R}_+^d$ then one may approximate solutions of (3.4) with the scheme

$$U_{\underline{j}}^{n+1} = U_{\underline{j}}^n - \Delta t \sum_{k=1}^{d} \frac{a_k}{\Delta x_k} \left(U_{\underline{j}}^n - U_{\underline{j}-\underline{e}_k}^n \right) \tag{3.9}$$

where $\underline{e}_k$ is the multi-index which is 1 in the jth component and 0 elsewhere. Treating $U_{\underline{j}}^n$ as a vector we may write this scheme generically as the matrix vector product $U_{\underline{j}}^{n+1} = AU_{\underline{j}}^n$ with A being the matrix associated with the discretisation. One may allow $a_k < 0$ for any $k \in \{1, \ldots, d\}$ and simply use a forwards spatial discretisation in the corresponding k terms within the sum.

Proposition 3.10. *The numerical scheme* (3.9) *with constant* $\boldsymbol{a} \in \mathbb{R}_+^d$ *for solving the* PDE (3.4) *with initial condition* $u(\boldsymbol{x}, 0) \in W^{2,\infty}(\mathbb{R}^d)$ *is pointwise convergent with order* 1 *if* $0 < \sum_{k=1}^{d} \frac{a_k \Delta t}{\Delta x_k} \leq 1$ *and* $U_{\underline{j}}^0 = u(x_{\underline{j}}, 0)$*. Further, with* $t_n = n\Delta t$ *and* $x_{\underline{j}} = (j_1 \Delta x_1, \ldots, j_d \Delta x_d)$ *one has the pointwise error bound*

$$\left| u(x_{\underline{j}}, t_n) - U_{\underline{j}}^n \right| \leq \frac{t_n}{2} \left(\Delta t \left\| \frac{\partial^2 u}{\partial t^2} \right\|_\infty + \sum_{k=1}^{d} a_k \Delta x_k \left\| \frac{\partial^2 u}{\partial x_k^2} \right\|_\infty \right).$$

Proof. Similar to the proof of Proposition 3.9 one may use Taylor series expansions about $u(x_{\underline{j}}, t_n)$ to obtain

$$\begin{aligned} \epsilon_{\underline{j}}^n = \left(1 - \sum_{k=1}^{d} \frac{a_k \Delta t}{\Delta x_k} \right) \epsilon_{\underline{j}}^{n-1} + \sum_{k=1}^{d} \frac{a_k \Delta t}{\Delta x_k} \epsilon_{\underline{j}-\underline{e}_k}^{n-1} \\ - \frac{\Delta t^2}{2} \frac{\partial^2}{\partial t^2} u(x_{\underline{j}}, \xi_t) + \sum_{k=1}^{d} \frac{a_k \Delta x_k \Delta t}{2} \frac{\partial^2}{\partial x_k^2} u(\xi_{x_k}, t_n), \end{aligned} \tag{3.10}$$

for some $\xi_{x_k} = (x_1, \ldots, x_{k-1}, \xi_k, x_{k+1}, \ldots, x_d)$ with $\xi_k \in [x_{j_k - 1}, x_{j_k}]$ for each $k = 1, \ldots, d$, and some $\xi_t \in [t_{n-1}, t_n]$. As before, with $0 \leq \sum_{k=1}^{d} \left| \frac{a_k \Delta t}{\Delta x_k} \right| \leq 1$ the contribution from the $\{\epsilon_j^{n-1}\}_j$ is a convex combination, and thus as $u(\boldsymbol{x}, t) = u(\boldsymbol{x} - \boldsymbol{a}t, 0)$ one has

$$|\epsilon_{\underline{j}}^n| \leq \max_{\underline{j}} |e_{\underline{j}}^{n-1}| + \frac{\Delta t^2}{2} \left\| \frac{\partial^2 u(\boldsymbol{x}, 0)}{\partial t^2} \right\|_\infty + \sum_{k=1}^{d} \frac{a_k \Delta x_k \Delta t}{2} \left\| \frac{\partial^2 u(\boldsymbol{x}, 0)}{\partial x_k^2} \right\|_\infty. \tag{3.11}$$

Now by recursively substituting the $\{\epsilon_{\underline{j}}^{n-1}\}_{\underline{j}}, \{\epsilon_{\underline{j}}^{n-2}\}_{\underline{j}}, \ldots$ terms into the equation and using $t_n = n\Delta t$ one obtains

$$|\epsilon_{\underline{j}}^n| \leq \max_{\underline{j}} |e_{\underline{j}}^0| + \frac{t_n}{2} \left(\Delta t \left\| \frac{\partial^2 u}{\partial t^2} \right\|_\infty + \sum_{k=1}^d a_k \Delta x_k \left\| \frac{\partial^2 u}{\partial x_k^2} \right\|_\infty \right) ,$$

from which it follows that the scheme is first order pointwise convergent. □

The derived error bound again tells us our scheme is first order in both the spatial and temporal variables. Whilst this pointwise bound is a nice result it is still insufficient for showing the convergence of the combination technique. Note that (3.10) gives an exact expression for the error over one time step. One may be tempted to substitute in the expression for the $\epsilon_{\underline{j}}^{n-1}$ in terms of the $\epsilon_{\underline{j}}^{n-2}$ and then iterate to obtain an exact expression of the error. However, this is somewhat cumbersome and leads to an accumulation of powers of $\frac{a_k \Delta t}{\Delta x_k}$ terms which do not fit with the classical error splitting model (2.22). In Section 3.3 we use a different approach to show that finite difference solutions satisfy the error splitting model.

Rather than the explicit upwind scheme, one could instead use a backwards time discretisation to obtain an implicit method. For example a backwards time backwards space (BTBS) discretisation of (3.2) leads to update formula

$$U_j^{n+1} + \frac{\Delta t}{\Delta x}(F_j^{n+1} - F_{j-1}^{n+1}) = U_j^n .$$

A von Neumann stability analysis shows this is unconditionally stable (for $\frac{\partial f}{\partial u} > 0$) and we may obtain an error bound similar to that of the forwards time scheme. Whilst implicit schemes typically require solving a linear system of equation we observe that if the left boundary condition is specified, e.g. $u(0, t) = g(t)$ for some $g : \mathbb{R} \to \mathbb{R}$, then each time step can be computed quickly via forward substitution. For periodic boundary conditions the full linear system must be solved. This scheme is also easily extended to the solution of the advection equation in higher dimensions. Reisinger [110] considered such an implicit method for the solution of the advection equation for $d \geq 1$ spatial dimensions and his work is discussed further in Section 3.3.

Second-order schemes

The Lax–Wendroff scheme [83] is second order in both space and time variables. It is typically presented as a two step method for the solution of the advection equation in one spatial dimension. The first step is given by

$$U_{j+1/2}^{n+1/2} = \frac{1}{2}\left(U_j^n + U_{j+1}^n\right) - \frac{\Delta t}{2\Delta x}\left(F_{j+1}^n - F_j^n\right) ,$$

and the second step by

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{\Delta x}\left(F_{j+1/2}^{n+1/2} - F_{j-1/2}^{n+1/2}\right) ,$$

with $F_{j\pm1/2}^{n+1/2} = f\left(U_{j\pm1/2}^{n+1/2}, x_{j\pm1/2}, t_{n+1/2}\right)$. If $f(u,x,t) = au$ with $a \in \mathbb{R}$ constant this reduces to the one step method

$$U_j^{n+1} = U_j^n - \frac{a\Delta t}{\Delta x}\left(\frac{U_{j+1}^n - U_{j-1}^n}{2} - \frac{a\Delta t}{2\Delta x}\left(U_{j+1}^n - 2U_j^n + U_{j-1}^n\right)\right) . \tag{3.12}$$

The pointwise error estimates for the FTBS scheme effectively used the monotonicity of the scheme (via the convex combination). Godunov's theorem [53] tells us that monotonicity is not possible for linear schemes of order more than 1. Thus it is more difficult to obtain pointwise error estimates in this case. Instead we will show stability condition via a von Neumann stability analysis to illustrate the approach.

Lemma 3.11. *The Lax–Wendroff scheme* (3.12) *for solving the* PDE (3.2) *with* $f(u) = au$ *and constant* $a \in \mathbb{R}$ *and initial condition* $u(x,0) \in L^2([0,1])$ *is stable for* $\left|\frac{a\Delta t}{\Delta x}\right| \leq 1$.

Proof. We consider a solution which consists of a single Fourier mode $k \in \mathbb{Z}$. Let $U_j^n = e^{rt}e^{\iota jk\Delta x}$ where $\iota := \sqrt{-1}$, then substituting into (3.12) and dividing out common term $e^{rt_n}e^{\iota jk\Delta x}$ we have

$$\begin{aligned} e^{r\Delta t} &= 1 - \frac{a\Delta t}{2\Delta x}\left(e^{\iota k\Delta x} - e^{-\iota k\Delta x}\right) + \frac{a^2\Delta t^2}{2\Delta x^2}\left(e^{\iota k\Delta x} - 2 + e^{-\iota k\Delta x}\right) \\ &= 1 - \iota\frac{a\Delta t}{\Delta x}\sin(k\Delta x) - \frac{a^2\Delta t^2}{\Delta x^2}\left(1 - \cos(k\Delta x)\right) . \end{aligned}$$

As stability requires that $|e^{r\Delta t}| \leq 1$ (von Neumann stability analysis) we have

$$\begin{aligned} 1 &\geq \left(1 - \frac{a^2\Delta t^2}{\Delta x^2}\left(1 - \cos(k\Delta x)\right)\right)^2 + \frac{a^2\Delta t^2}{\Delta x^2}\sin^2(k\Delta x) \\ &= 1 - \frac{a^2\Delta t^2}{\Delta x^2}\left(1 - \frac{a^2\Delta t^2}{\Delta x^2}\right)\left(1 - \cos(k\Delta x)\right)^2 . \end{aligned}$$

As $0 \leq 1 - \cos(k\Delta x) \leq 2$ then stability is satisfied if $0 \leq 4\frac{a^2\Delta t^2}{\Delta x^2}\left(1 - \frac{a^2\Delta t^2}{\Delta x^2}\right) \leq 1$. Thus if $-1 \leq \frac{a\Delta t}{\Delta x} \leq 1$ we have $|U_j^{n+1}| = |e^{r\Delta t}U_j^n| \leq |U_j^n|$. As the chosen Fourier mode was arbitrary then $|U_j^{n+1}| \leq |U_j^n|$ for solutions consisting of any Fourier mode. It follows that the problem is stable for $|\frac{a\Delta t}{\Delta x}| \leq 1$ given any $u(x,0) \in L^2([0,1])$ via its Fourier decomposition and the linearity of the discretisation. $\square$

We can also check the order of consistency for the Lax–Wendroff scheme.

Lemma 3.12. *The Lax–Wendroff scheme* (3.12) *for solving the* PDE (3.2) *with* $f(u) = au$ *and constant* $a \in \mathbb{R}$ *is second order consistent if* $u(x, 0) \in C^3([0, 1])$.

Proof. The exact solution clearly satisfies $u(x, t) \in C^3([0, 1] \times [0, \infty))$, now consider the numerical scheme applied to the exact solution, that is

$$\begin{aligned} u(x, t + \Delta t) - u(x, t) + \frac{a\Delta t}{2\Delta x}(u(x + \Delta x, t) - u(x - \Delta x, t)) \\ - \frac{a^2\Delta t^2}{2\Delta x^2}(u(x + \Delta x, t) - 2u(x, t) + u(x - \Delta x, t)) \,, \end{aligned}$$

which we expect to equal zero if the scheme is exact. Now substituting the Taylor expansions

$$\begin{aligned} u(x, t + \Delta t) &= u(x, t) + \Delta t\frac{\partial u(x, t)}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u(x, t)}{\partial t^2} \\ &\quad + \int_0^{\Delta t} \frac{(\Delta t - \tau)^2}{2}\frac{\partial^3 u(x, t + \tau)}{\partial t^3}\, d\tau \\ &= u(x, t) + \Delta t\frac{\partial u(x, t)}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 u(x, t)}{\partial t^2} + \mathcal{O}(\Delta t^3) \end{aligned}$$

and

$$\begin{aligned} u(x + \Delta x, t) &= u(x, t) + \Delta x\frac{\partial u(x, t)}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 u(x, t)}{\partial x^2} \\ &\quad + \int_0^{\Delta x} \frac{(\Delta x - y)^2}{2}\frac{\partial^3 u(x + y, t)}{\partial t^3}\, dy \\ &= u(x, t) + \Delta x\frac{\partial u(x, t)}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 u(x, t)}{\partial x^2} + \mathcal{O}(\Delta x^3) \end{aligned}$$

one obtains

$$\begin{aligned} \Delta t\frac{\partial u(x, t)}{\partial t} + \Delta t^2\frac{\partial^2 u(x, t)}{\partial t^2} + \mathcal{O}(\Delta t^3) + a\Delta t\frac{\partial u(x, t)}{\partial x} + \mathcal{O}(\Delta t\Delta x^2) \\ - a^2\Delta t^2\frac{\partial^2 u(x, t)}{\partial x^2} + \mathcal{O}(\Delta t^2\Delta x) \end{aligned}$$

Now using the fact that $\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0$ and thus also $\frac{\partial^2 u}{\partial t^2} = a^2\frac{\partial^2 u}{\partial x^2}$ then the partial derivatives cancel out and one has

$$\begin{aligned} & u(x, t + \Delta t) - u(x, t) + \frac{a\Delta t}{2\Delta x}(u(x + \Delta x, t) - u(x - \Delta x, t)) \\ & -\frac{a^2\Delta t^2}{2\Delta x^2}(u(x+\Delta x, t)-2u(x, t)+u(x-\Delta x, t)) = \mathcal{O}(\Delta t^3)+\mathcal{O}(\Delta t^2\Delta x)+\mathcal{O}(\Delta t\Delta x^2) \,. \end{aligned}$$

Thus the Lax–Wendroff scheme is second order consistent. □

It follows from the Lax–Richtmyer equivalence theorem that the scheme is second order convergent if $u(x,0) \in H^3([0,1])$.

Extending the Lax–Wendroff method to higher dimensions is not straightforward and indeed the naive extension to two spatial dimensions

$$\begin{aligned} u^{n+1}_{j_1,j_2} = u^n_{j_1,j_2} &+ \frac{a_1\Delta t}{\Delta x_1}\left(\frac{u^n_{j_1+1,j_2} - u^n_{j_1-1,j_2}}{2} + \frac{a\Delta t}{2\Delta x_1}\left(u^n_{j_1+1,j_2} - 2u^n_{j_1,j_2} + u^n_{j_1-1,j_2}\right)\right) \\ &+ \frac{a_2\Delta t}{\Delta x_2}\left(\frac{u^n_{j_1,j_2+1} - u^n_{j_1,j_2-1}}{2} + \frac{a\Delta t}{2\Delta x_2}\left(u^n_{j_1,j_2+1} - 2u^n_{j_1,j_2} + u^n_{j_1,j_2-1}\right)\right) \end{aligned}$$

can be shown to be (unconditionally) unstable. For a problem with constant velocity $\boldsymbol{a} \in \mathbb{R}^d$ in two spatial dimensions the Lax–Wendroff method is given by

$$\begin{aligned} u^{n+1}_{j_1,j_2} = u^n_{j_1,j_2} &- \frac{a_1\Delta t}{2\Delta x_1}(u^n_{j_1+1,j_2} - u^n_{j_1-1,j_2}) - \frac{a_2\Delta t}{2\Delta x_2}(u^n_{j_1,j_2+1} - u^n_{j_1,j_2-1}) \\ &+ \frac{a_1^2\Delta t^2}{2\Delta x_1^2}(u^n_{j_1+1,j_2} - 2u^n_{j_1,j_2} + u^n_{j_1-1,j_2}) \\ &+ \frac{a_2^2\Delta t^2}{2\Delta x_2^2}(u^n_{j_1,j_2+1} - 2u^n_{j_1,j_2} + u^n_{j_1,j_2-1}) \\ &+ \frac{a_1a_2\Delta t^2}{4\Delta x_1\Delta x_2}(u^n_{j_1+1,j_2+1} - u^n_{j_1-1,j_2+1} - u^n_{j_1+1,j_2-1} + u^n_{j_1-1,j_2-1})\,. \end{aligned}$$

Notice the additional term which approximates the mixed derivative $\frac{\partial^2 u}{\partial x_1 \partial x_2}$ which was not present in the naive extension. Turkel [122] shows that this scheme is stable iff $(a_1\Delta t/\Delta x_1)^{3/2} + (a_2\Delta t/\Delta x_2)^{3/2} \leq 1$. An example of the Lax–Wendroff scheme applied to an advection problem on $[0,1]$ with periodic boundaries is depicted in Figure 3.2.

For the advection problem with more than two spatial dimensions many 2nd order schemes have been studied but it is not particularly clear if any one of them could be said to be the definitive Lax–Wendroff scheme in a given number of dimensions. Fortunately another approach may be used to extend one dimensional schemes to higher dimensional domains, so called directional splitting (also as dimensional splitting and more generally operator splitting). Directional splitting is based upon splitting methods developed for the solutions of differential equations [98]. We illustrate the approach on the advection equation in two spatial dimensions

$$\frac{\partial u}{\partial t} + a_1\frac{\partial u}{\partial x_1} + a_2\frac{\partial u}{\partial x_2} = 0$$

with constants $a_1, a_2 \in \mathbb{R}$ and $u = u(x_1,x_2,t)$. If u is analytic for all $t \geq 0$ we may write

$$u(\cdot,\cdot,t+\Delta t) = \sum_{k=0}^{\infty}\frac{\Delta t^k}{k!}\frac{\partial^k}{\partial t^k}u(\cdot,\cdot,t) =: e^{\Delta t\frac{\partial}{\partial t}}u(\cdot,\cdot,t)\,.$$

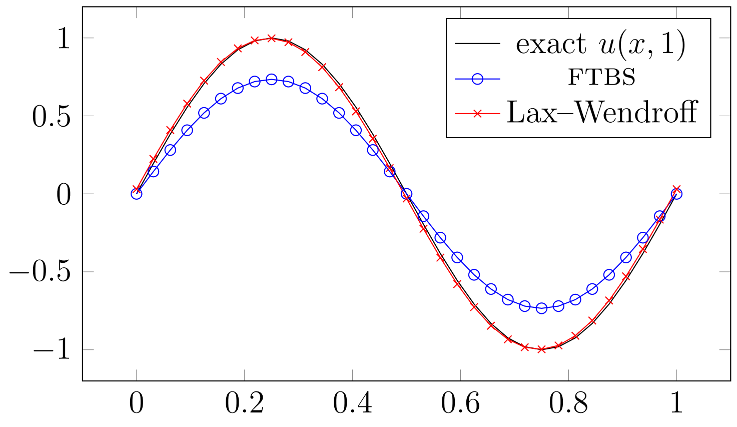


Figure 3.2: *Here we compare the* FTBS *and Lax–Wendroff schemes applied to the advection problem* $\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0$ *on the domain* $[0,1]$ *with initial condition* $u(x,0) = \sin(2\pi x)$ *and periodic boundary conditions. The exact solution at time* $t = 1$ *satisfies* $u(x,1) = u(x,0)$*. Applying the two schemes to this problem with* $\Delta x = 2^{-5}$ *and* $\Delta t = 2^{-6}$ *we observe the* FTBS *solution has suffered from some numerical diffusion whilst the Lax–Wendroff scheme is much more accurate but exhibits some slight dispersion.*

Now substituting the advection equation one has

$$u(\cdot,\cdot,t+\Delta t) = e^{\Delta t\left(-a_1\frac{\partial}{\partial x_1} - a_2\frac{\partial}{\partial x_2}\right)} u(\cdot,\cdot,t) = e^{\Delta t\left(-a_1\frac{\partial}{\partial x_1}\right)} e^{\Delta t\left(-a_2\frac{\partial}{\partial x_2}\right)} u(\cdot,\cdot,t)\,,$$

where $e^{\Delta t\left(-a_1\frac{\partial}{\partial x_1}\right)} := \sum_{k=0}^{\infty} \frac{(-a_1\Delta t)^k}{k!}\frac{\partial^k}{\partial x_1^k}$, similarly for $e^{\Delta t\left(-a_1\frac{\partial}{\partial x_1}\right)}$, and the last equality holds here as the operators $-a_1\frac{\partial}{\partial x_1}$ and $-a_2\frac{\partial}{\partial x_2}$ commute. Now we may introduce the new initial value problems

$$\tilde{u}(\cdot,\cdot,t+\Delta t) = e^{\Delta t\left(-a_2\frac{\partial}{\partial x_2}\right)} u(\cdot,\cdot,t) \quad \text{with} \quad \tilde{u}(\cdot,\cdot,t) = u(\cdot,\cdot,t)\,,$$

and

$$u(\cdot,\cdot,t+\Delta t) = e^{\Delta t\left(-a_1\frac{\partial}{\partial x_1}\right)} \tilde{u}(\cdot,\cdot,t+\Delta t) \quad \text{with} \quad u(\cdot,\cdot,t) = \tilde{u}(\cdot,\cdot,t+\Delta t)\,.$$

Thus, we can solve our two dimensional advection problem by treating it as two separate one dimensional advection problems. In particular we can first step $u(t)$ over the x_2 dimension to obtain $\tilde{u}$, and then step $\tilde{u}$ forward over the x_1 dimension to obtain the updated $u(t+\Delta t)$. For the scalar advection equation with constant velocity we can in fact evolve from time 0 to t over each direction separately. For more complex problems this serves as an approximation by treating coefficients as locally constant and thus one applies each scheme one time step at a time. In this latter case the discretisation is effectively a tensor product of the one dimensional discretisations. This concept extends naturally to higher dimensions and means one can use a one dimensional scheme to solve the d-dimensional problem. In general such directional splittings do not give the same result as a 'full' d-dimensional scheme, that is additional approximation errors are introduced. For example, given an operator $C = A + B$ then in general $e^{\Delta t C} \neq e^{\Delta t A} e^{\Delta t B}$, in fact it can be shown [75] that

$$\left(e^{\Delta t A} e^{\Delta t B} - e^{\Delta t(A+B)}\right) u(\cdot,\cdot,t) = \frac{\Delta t^2}{2}(AB - BA)u(\cdot,\cdot,t) + \mathcal{O}(\Delta t^3) \qquad (3.13)$$

and thus $e^{\Delta t A} e^{\Delta t B}$ is a first order consistent approximation to $e^{\Delta t C}$. If A, B commute then the RHS of (3.13) vanishes and the splitting does not introduce any additional error into the problem as in our example of the scalar advection problem with constant velocity. A second order approximation may be obtained for general problems via the Strang splitting

$$e^{\frac{\Delta t}{2} A} e^{\Delta t B} e^{\frac{\Delta t}{2} A} - e^{\Delta t(A+B)} = \mathcal{O}(\Delta t^3)\,.$$

Higher order splittings also exist, see [129, 75]. For example, a fourth order splitting is given by

$$e^{\frac{c}{2}\Delta t A} e^{c\Delta t B} e^{\frac{1-2^{1/3}}{2} c\Delta t A} e^{-2^{1/3} c\Delta t B} e^{\frac{1-2^{1/3}}{2} c\Delta t A} e^{c\Delta t B} e^{\frac{c}{2}\Delta t A}$$

with $c = (2 - 2^{1/3})^{-1}$. Additionally, splittings can be extended to more than 2 terms. For example given $\Delta t D = \Delta t(A+B+C)$ one has the second order Strang splitting

$$e^{\frac{\Delta t}{2} A} e^{\frac{\Delta t}{2} B} e^{\Delta t C} e^{\frac{\Delta t}{2} B} e^{\frac{\Delta t}{2} A}\,.$$

For a detailed account of splitting methods applied to PDEs we refer the reader to [75, 39, 85].

Remark 3.13. As already noted, for the scalar advection problem with constant velocity the operators $a_k \frac{\partial}{\partial x_k}$ commute and thus directional splittings are exact, that is $e^{\Delta t \frac{\partial}{\partial t}} = \prod_{k=1}^{d} e^{-a_k \Delta t \frac{\partial}{\partial x_k}}$. This means directional splittings work exceptionally well for advection problems. Further, we have discussed how the directional splitting is equivalent to a tensor product discretisation for linear problems. For example, the directional splitting concept is similar to the unidirectional principle used in tensor product quadrature formula. This would suggest that such approaches may work very well with the combination technique. In essence, if a finite difference discretisation in one dimension satisfied an error splitting then it would be reasonable to expect that when applied to higher dimensions via directional splittings then approximations satisfy the higher dimensional error splitting.

Remark 3.14. Another advantage of using directional splitting methods is one need only satisfy stability for each one dimensional problem. For example, when applied to the Lax–Wendroff discretisation in 2 spatial dimensions we require Δt such that $|a_1|\Delta t \leq \Delta x_1$ and $|a_2|\Delta t \leq \Delta x_2$ as opposed to $(a_1\Delta t/\Delta x_1)^{3/2} + (a_2\Delta t/\Delta x_2)^{3/2} \leq 1$ for the full 2D scheme.

Crank-Nicolson is an implicit method originally developed to solve the heat equation $\frac{\partial}{\partial t}u - D\frac{\partial^2}{\partial x^2}u = 0$. It can be derived starting from a centred difference approximation about $U_j^{n+\frac{1}{2}}$ leading to

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = D\frac{U_{j-1}^{n+\frac{1}{2}} - 2U_j^{n+\frac{1}{2}} + U_{j+1}^{n+\frac{1}{2}}}{\Delta x^2}\,.$$

On the right hand side we simply approximate each $U_j^{n+\frac{1}{2}}$ term with the average $\frac{1}{2}(U_j^n + U_j^{n+1})$ leading to the equations

$$U_j^{n+1} - \frac{D\Delta t}{2\Delta x^2}(U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}) = U_j^n + \frac{D\Delta t}{2\Delta x^2}(U_{j-1}^n - 2U_j^n + U_{j+1}^n)\,.$$

A von Neumann analysis shows that this is unconditionally stable. One can therefore use relatively large time steps compared to the typical restriction of $\Delta t \propto \Delta x^2$ for explicit methods for the heat equation.

The Crank-Nicolson scheme can be adapted to the advection equation replacing the centred difference approximation of $\frac{\partial^2}{\partial x^2}u$ with a centred difference approximation of $\frac{\partial}{\partial x}u$. This leads to the equation

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{F_{j+1}^{n+\frac{1}{2}} - F_{j-1}^{n+\frac{1}{2}}}{2\Delta x} = 0$$

and upon approximating the $F_j^{n+\frac{1}{2}}$ terms with the average $\frac{1}{2}(F_j^n + F_j^{n+1})$ we obtain

$$U_j^{n+1} + \frac{\Delta t}{4\Delta x}(F_{j+1}^{n+1} - F_{j-1}^{n+1}) = U_j^n - \frac{\Delta t}{4\Delta x}(F_{j+1}^n - F_{j-1}^n)\,. \tag{3.14}$$

It is readily checked that this too is unconditionally stable and gives a second order approximation in both space and time. The unconditional stability allows one to use arbitrarily large time steps such that despite the increased cost associated with solving the linear system given by (3.14) one could potentially obtain solutions in a similar time. Unfortunately, in practice using time steps significantly larger than the CFL condition typically leads to large errors in the approximation. Thus it is typically much cheaper to use an explicit method to obtain an approximate solution having some desired error. For this reason we will not consider implicit schemes like Crank-Nicolson in further detail for pure advection problems.

The Crank-Nicolson scheme is quite effective for advection-diffusion problems. Consider the PDE (3.5) in one spatial dimension with constants $\boldsymbol{a} = a \in \mathbb{R}$ and $D > 0$, then one has the numerical scheme

$$\begin{aligned} u_j^{n+1} &+ \frac{a\Delta t}{4\Delta x}(u_{j+1}^{n+1} - u_{j-1}^{n+1}) - \frac{D\Delta t}{\Delta x^2}(u_{j+1}^{n+1} - u_j^{n+1} + u_{j-1}^{n+1}) \\ &= u_j^n - \frac{a\Delta t}{4\Delta x}(u_{j+1}^n - u_{j-1}^n) + \frac{D\Delta t}{\Delta x^2}(u_{j+1}^n - u_j^n + u_{j-1}^n)\,, \end{aligned} \tag{3.15}$$

which is unconditionally stable and gives a second order approximation with respect to Δx and Δt. The unconditional stability here means we can have $2D\Delta t \gg \Delta x^2$ unlike typical explicit methods for advection diffusion equations. However, for accuracy in the advection term we still may choose $a\Delta t \propto \Delta x$. This scheme extends naturally to problems in higher dimensions, for $d \geq 1$ and $\boldsymbol{a} \in \mathbb{R}^d$ constant one has the scheme

$$\begin{aligned} u_{\underline{j}}^{n+1} + \sum_{k=1}^{d} &\left(\frac{a_k\Delta t}{4\Delta x_k}(u_{\underline{j}+\underline{e}_k}^{n+1} - u_{\underline{j}-\underline{e}_k}^{n+1}) - \frac{D\Delta t}{\Delta x_k^2}(u_{\underline{j}+\underline{e}_k}^{n+1} - u_{\underline{j}}^{n+1} + u_{\underline{j}-\underline{e}_k}^{n+1}) \right) \\ &= u_{\underline{j}}^n - \sum_{k=1}^{d} \left(\frac{a_k\Delta t}{4\Delta x_k}(u_{\underline{j}+\underline{e}_k}^n - u_{\underline{j}-\underline{e}_k}^n) - \frac{D\Delta t}{\Delta x_k^2}(u_{\underline{j}+\underline{e}_k}^n - u_{\underline{j}}^n + u_{\underline{j}-\underline{e}_k}^n) \right) \,. \end{aligned}$$

Mixed order schemes

Here we briefly mention some advection schemes which have a higher order of temporal accuracy than spatial accuracy. Such schemes will be useful for numerical experiments that involve extrapolation of spatial error terms and are fundamental to the practical application of results obtained in Section 3.3. Consider the one dimensional scalar advection equation with a centred difference approximation of the spatial derivative, that is with $a \in \mathbb{R}$

$$\frac{\partial U_j^n}{\partial t} = -\frac{a}{2\Delta x}\left(U_{j+1}^n - U_{j-1}^n\right) \,.$$

It is well known that if the $\frac{\partial U_j^n}{\partial t}$ is approximated as $\frac{1}{\Delta t}\left(U_j^{n+1} - U_j^n\right)$ then one obtains an unconditionally unstable scheme. However, if $\frac{\partial U_j^n}{\partial t}$ is approximated with a classical Runge–Kutta method of order 3 or higher then the scheme is stable for sufficiently small Δt. For example, the classical 4th order Runge–Kutta method for approximating $\frac{\partial u(x,t)}{\partial t} = f(t, u(x,t))$ is given by

$$u(x, t + \Delta t) \approx u(x,t) + \frac{\Delta t}{6}(k_1(x,t) + 2k_2(x,t) + 2k_3(x,t) + k_4(x,t))$$

where

$$\begin{aligned}
k_1(x,t) &= f(t, u(x,t)) \\
k_2(x,t) &= f\left(t + \frac{\Delta t}{2}, u(x,t) + \frac{\Delta t}{2} k_1(x,t)\right) \\
k_3(x,t) &= f\left(t + \frac{\Delta t}{2}, u(x,t) + \frac{\Delta t}{2} k_2(x,t)\right) \\
k_4(x,t) &= f\left(t + \Delta t, u(x,t) + \Delta t k_3(x,t)\right) .
\end{aligned}$$

With $f(t, u(x,t)) = -\frac{a}{2\Delta x}(u(x+\Delta x, t) - u(x-\Delta x, t))$ then the resulting scheme for solving the advection equation is second order with respect to Δx, fourth order with respect to Δt, and stable if $|a|\Delta t < 2.82\Delta x$ [75]. If the 2nd order spatial errors were successfully cancelled with an extrapolation method applied to this scheme for $u \in C^4(\Omega)$ then one would expect to obtain a fourth order scheme (as the centred approximation of the spatial derivative means there are no odd order terms in the error expansion). This scheme can be applied to higher dimensional problems via directional splitting or via finite difference discretisations described in Section 3.3.

Other schemes

We briefly remark here that for hyperbolic conservation laws (3.2) there has been much research into discretisations which satisfy additional stability properties. Among these are weighted essentially non-oscillatory (WENO) schemes and strong stability preserving (SSP) Runge–Kutta methods, see for example [78]. Whilst this is a very interesting area of research with such schemes being essential for the practical numerical simulation of many physical problems we will not investigate these in this thesis. As our focus is on the application of the combination technique to approximate solutions of hyperbolic PDEs we use much simpler schemes for which the approximation error is better understood. Further, it is not clear if the additional stability properties of more complex schemes would be preserved by the application of the combination technique. Such schemes often rely on convex combinations but as the classical combination technique is not convex it would likely fail to preserve these additional stability properties. Investigating the compatibility of the combination technique with such schemes would be an interesting direction of research.

3.3 Solving with the Combination Technique

When PDEs involving time evolution are solved with the combination technique one evolves the solution on many anisotropic grids to some fixed time t and then combines the results accordingly. The application of the sparse grid combination technique applied to the advection equation has been studied by Lastdrager et.al. [82] and Reisinger [110]. Reisinger studied an implicit first order finite difference solution of the advection equation in arbitrary dimensions. He showed that the implicit first order finite difference solutions when interpolated by continuous basis functions (piecewise linear for example) satisfy the error splitting model. This was done by studying the error of several corresponding semi-discrete problems whereby one discretises a subset of the spatial dimensions and supposes that the operator is exact along the remaining dimensions. The unconditional stability of the implicit method allowed Reisinger to apply the combination technique to a space-time sparse grid. Using the error splitting he was able to obtain error bounds for the combined solution. Poisson and advection-diffusion problems were also considered. It was suggested that the semi-discrete framework could be used to study different numerical schemes for advection and/or diffusion and perhaps even other problems entirely. For example, one might consider extending Reisinger's bounds on the advection equation to a second order method. Crank-Nicolson is another implicit method for which the space-time sparse grid could also be considered. However, extending Reisinger's work to this scheme is difficult because it relies on a discrete maximum principle. As Godunov's theorem [53] implies linear discretisations with order greater than 1 can not have the property that new extrema are not generated then one is unable to obtain a similar discrete maximum result for the Crank-Nicolson method. The extension of this work to different discretisations remains open and is not something that will be considered in this thesis.

Lastdrager et.al. considered the solution to the advection equation in two spatial dimensions discretised using the method of lines (MOL) with finite difference approximations to spatial derivatives and gave numerical results focusing on some specific explicit methods. Several assumptions were made to simplify the analysis. First, it was assumed that the error from time stepping is negligible compared to the spatial discretisation error. Second, it was also assumed that the approximation error obtained from interpolation of component grids onto a full (or sparse) grid is also negligible. This is justified by their focus on numerical results where a 3rd order upwind biased discretisation of the spatial derivatives is used with 4th order Runge–Kutta time integration and 4th order interpolation (referred to as prolongation in their work). The authors then considered how the leading order error terms are extrapolated when the combination technique is applied and obtained an estimate of the combination of the leading order error term from different component grids. The effect of several combinations throughout the computation was also studied and they show that M combinations throughout the computation leads to a decrease of the leading order error terms by factors M^{-1}.

Error bounds are common in the literature for finite difference solutions of time-dependent PDEs, see for example [75, 88], but to obtain reasonable estimates when the combination technique is applied we need something stronger than a bound. We require an equality similar to the error splittings studied in Section 2.2. This can be difficult to establish for a specific PDE and discretisation in which case one might settle for an analysis of the leading order error terms. Here we extend the study of the leading error terms of the spatial discretisation error of a two dimensional scalar advection problem by Lastdrager et.al. [82] to a precise analysis of the scalar advection problem in arbitrary dimensions. As in this reference literature we study the MOL approach using finite differences to discretise the spatial derivatives. Consider solutions to the advection equation

$$\frac{\partial u}{\partial t} + \boldsymbol{a} \cdot \nabla u = 0\,, \tag{3.16}$$

with constant $\boldsymbol{a} \in \mathbb{R}^d$, periodic boundary conditions and periodic initial condition $u(\boldsymbol{x}, 0) := u_0(\boldsymbol{x}) \in H^1_{\text{per}}([0,1]^d)$ (where $H^1_{\text{per}}([0,1]^d)$ consists of functions $f \in C(\mathbb{R}^d)$ such that $f(\boldsymbol{x}) = f(\boldsymbol{x} + \boldsymbol{e}_k)$ for all $\boldsymbol{x}$ with $\boldsymbol{e}_k$ being the unit vector in the kth direction, and f restricted to $[0,1]^d$ is an element of $H^1([0,1])^d$, that is $\sum_{|\underline{i}|\leq 1} \|D^{\underline{i}} f\|^2_{L_2([0,1]^d)}$ is finite). For $t \geq 0$ the exact solution $u(\boldsymbol{x}, t) = u_0(\boldsymbol{x} - \boldsymbol{a}t)$ is obtained via the method of characteristics. Observe that the solution is in the same function space as the initial condition as the norm is invariant under translation. Establishing the accuracy of numerical solutions typically requires additional assumptions on differentiability of the initial condition which we will come back to later. Note that we may represent u_0 by the Fourier series

$$u_0(\boldsymbol{x}) = \sum_{\boldsymbol{\xi} \in \mathbb{Z}^d} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota \boldsymbol{\xi} \cdot \boldsymbol{x}}\,,$$

where the $\hat{u}_{0,\boldsymbol{\xi}}$ are the Fourier coefficients

$$\hat{u}_{0,\boldsymbol{\xi}} = \int_{[0,1]^d} u_0(\boldsymbol{x}) e^{2\pi\iota \boldsymbol{\xi} \cdot \boldsymbol{x}}\, d\boldsymbol{x}\,.$$

Substituting in the characteristics one has the exact solution

$$u(\boldsymbol{x},t) = \sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota\boldsymbol{\xi}\cdot(\boldsymbol{x}-\boldsymbol{a}t)} .$$

Using operator notation we may write $u(\boldsymbol{x},t) = e^{-t\boldsymbol{a}\cdot\nabla}u_0(\boldsymbol{x})$ for which we observe the operator $e^{-t\boldsymbol{a}\cdot\nabla}$ has the eigenvalues $e^{-2\pi\iota\boldsymbol{\xi}\cdot\boldsymbol{a}t}$ and corresponding eigenfunctions $e^{-2\pi\iota\boldsymbol{\xi}\cdot\boldsymbol{x}}$ for each $\boldsymbol{\xi}\in\mathbb{Z}^d$.

We now consider a discretisation of the ∇ operator with a finite difference approximation. For $m\in\mathbb{Z}$, $i\in\mathbb{N}$ and $k\in\{1,\dots,d\}$ let $S_{k,i,m}$ be the *shift operator* defined by $S_{k,i,m}u(\boldsymbol{x},t) := u(\boldsymbol{x}+m2^{-i}\boldsymbol{e}_k,t)$ with $\boldsymbol{e}_k$ being the unit vector in the kth direction. Now consider the approximation of ∇ in the advection equation with the operator $D_{\underline{i}} = (D_{i_1},\dots,D_{i_d})$ with each D_{i_k} a superposition of shift operators in the kth direction. In particular, let D_{i_k} approximate $\frac{\partial}{\partial x_k}$ via the shift operators $S_{k,i_k,m}$ with $m=-r,-r+1,\dots,r-1,r$ for some fixed integer $r>0$. We define

$$D_{i_k}u(\boldsymbol{x},t) = 2^{i_k}\sum_{m=-r}^{r}\alpha_{k,m}S_{k,i_k,m}u(\boldsymbol{x},t)\,, \tag{3.17}$$

where $\alpha_{k,m}\in\mathbb{R}$ are some appropriate coefficients.

Lemma 3.15. *If D_{i_k} is a pth order consistent approximation of $\frac{\partial}{\partial x_k}$ then*

$$\sum_{m=-r}^{r}\alpha_{k,m}m^q = \begin{cases} 1 & \textit{if } q=1 \\ 0 & \textit{if } q=0,2,3,\dots,p. \end{cases} \tag{3.18}$$

Proof. If D_{i_k} is a pth order approximation of $\frac{\partial}{\partial x_k}$ then it is exact for polynomials of degree p or less. Now as both D_{i_k} and $\frac{\partial}{\partial x_k}$ are translation invariant it is enough to consider approximations at $x_k=0$. Consider the monomial x_k^q with $q\in\mathbb{N}$, then one has

$$D_{i_k}x_k^q|_{x_k=0} = 2^{i_k}\sum_{m=-r}^{r}\alpha_{k,m}(x_k+m2^{-i_k})^q\Big|_{x_k=0} = 2^{i_k(1-q)}\sum_{m=-r}^{r}\alpha_{k,m}m^q\,.$$

For comparison one has

$$\frac{\partial}{\partial x_k}x_k^q\Big|_{x_k=0} = qx_k^{q-1}\big|_{x_k=0} = \begin{cases} 1 & \text{if } q=1 \\ 0 & \text{otherwise.} \end{cases}$$

Thus if D_{i_k} is exact for monomials up to degree p then clearly (3.18) holds. As any polynomial of degree p or less can be expressed as a superposition of monomials up to degree p one has the desired result. □

Let $\omega_{\underline{i}} = \omega_{\underline{i}}(\boldsymbol{x}, t)$ be the solution of

$$\frac{\partial \omega}{\partial t} + \boldsymbol{a} \cdot D_{\underline{i}} \omega = 0\,, \tag{3.19}$$

with initial condition $\omega(\boldsymbol{x}, 0) = u_0(\boldsymbol{x})$. If we restrict ourselves to considering the function ω on the grid points $\Omega_{\underline{i}} \subset [0,1]^d$ (with the usual definition $\Omega_{\underline{i}} = \Omega_{i_1} \times \cdots \times \Omega_{i_d}$ and $\Omega_l := \{s2^{-l} : s = 0, 1, \dots, 2^l\}$) then we observe that the shift operators making up the D_{i_k} translate each grid point in $\Omega_{\underline{i}}$ to another grid point in $\Omega_{\underline{i}}$. Therefore we may write $-\boldsymbol{a} \cdot D_{\underline{i}} \omega(\Omega_{\underline{i}}, t)$ generically as a matrix vector product $A_{\underline{i}} \omega_{\underline{i}}$ where $\omega_{\underline{i}} = \omega(\Omega_{\underline{i}}, t)$. Thus our discretised PDE restricted to $\omega_{\underline{i}} := \omega(\Omega_{\underline{i}})$ reduces to a linear system of ODE's

$$\frac{\partial \omega_{\underline{i}}}{\partial t}(t) = A_{\underline{i}} \omega_{\underline{i}}(t)$$

which has solution $\omega_{\underline{i}}(t) = \exp(tA_{\underline{i}})\omega_{\underline{i}}(0)$ where $\exp(tA_{\underline{i}}) := \sum_{n=0}^{\infty} \frac{t^n A_{\underline{i}}^n}{n!}$ is the usual matrix exponential. For $d = 1$, $A_{\underline{i}}$ is a circulant matrix which is diagonalised via the discrete Fourier transform (DFT). For $d = 2$, $A_{\underline{i}}$ is a block circulant matrix with circulant blocks (BCCB matrix) which is similarly diagonalised via the 2-dimensional DFT. This structure is similarly extended to higher dimensions. For a stable numerical scheme one typically requires that the finite difference discretisation $D_{\underline{i}}$ is such that the eigenvalues $\lambda_{\boldsymbol{\xi}}$ of the matrix $A_{\underline{i}}$ have non-positive real part (that is $\mathrm{Re}(\lambda_{\boldsymbol{\xi}}) \leq 0$) in which case it follows that $\|\exp(tA_{\underline{i}})\|_2 \leq 1$. We are interested in the difference between the approximation $\omega(\boldsymbol{x}, t)$ and the exact solution $u(\boldsymbol{x}, t)$ not only on the grid points $\Omega_{\underline{i}}$ but everywhere on $[0,1]^d$. Therefore, despite the discretised operator corresponding to a grid $\Omega_{\underline{i}}$, we still consider $\omega(\boldsymbol{x}, t)$ as a continuous function over $[0,1]^d$. This can be thought of as a continuous family of solutions on grids $\Omega_{\underline{i}}$ translated by some $\boldsymbol{y}$ with $0 \leq y_k \leq 2^{-i_k}$ for each $k \in \{1, \dots, d\}$. Note that as we have defined the $D_{\underline{i}}$ in terms of shift operators (i.e. translations) they are perfectly valid operators on continuous function spaces like L^2. The following lemma gives an expression for the solution $\omega(\boldsymbol{x}, t)$ via the Fourier series of the initial condition.

Lemma 3.16. *Suppose $u_0(\boldsymbol{x}) \in L^2_{per}([0,1]^d)$ has the Fourier series*

$$u_0(\boldsymbol{x}) = \sum_{\boldsymbol{\xi} \in \mathbb{Z}^d} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota(\boldsymbol{\xi}\cdot\boldsymbol{x})}\,,$$

then a unique solution $\omega(\boldsymbol{x}, t)$ to (3.19) *with initial condition $\omega(\boldsymbol{x}, 0) = u_0(\boldsymbol{x})$ exists and is given by*

$$\omega(\boldsymbol{x}, t) = \sum_{\boldsymbol{\xi} \in \mathbb{Z}^d} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota(\boldsymbol{\xi}\cdot\boldsymbol{x}) + c_{\boldsymbol{\xi}} t}\,.$$

where

$$c_{\boldsymbol{\xi}} = -\sum_{k=1}^{d} a_k 2^{i_k} \sum_{m=-r}^{r} \alpha_{k,i_k,m} e^{2\pi \imath m 2^{-i_k} \xi_k} . \tag{3.20}$$

Further, if $\Re(c_\xi) \leq 0$ *for all* $\boldsymbol{\xi} \in \mathbb{Z}^d$ *then given any* $t \geq 0$ *one has* $\omega(\boldsymbol{x}, t) \in L^2_{per}([0,1]^d)$*.*

Proof. Suppose that $\omega(\boldsymbol{x}, t)$ has the form

$$\omega(\boldsymbol{x}, t) = \sum_{\boldsymbol{\xi} \in \mathbb{Z}^d} b_{\boldsymbol{\xi}} e^{2\pi\imath(\boldsymbol{\xi}\cdot\boldsymbol{x})} e^{c_{\boldsymbol{\xi}} t} .$$

Note that we have $\omega(\boldsymbol{x}, 0) = u_0(\boldsymbol{x})$ and thus $b_{\boldsymbol{\xi}} = \hat{u}_{0,\boldsymbol{\xi}}$ for all $\boldsymbol{\xi} \in \mathbb{Z}^d$. As the PDE (3.19) is linear we may consider each component of the Fourier series separately. Fixing $\boldsymbol{\xi} \in \mathbb{Z}^d$, for the series to satisfy (3.19) we require

$$\begin{aligned} 0 &= c_{\boldsymbol{\xi}} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\imath(\boldsymbol{\xi}\cdot\boldsymbol{x})} e^{c_{\boldsymbol{\xi}} t} + \boldsymbol{a} \cdot D_{\underline{i}} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\imath(\boldsymbol{\xi}\cdot\boldsymbol{x})} e^{c_{\boldsymbol{\xi}} t} \\ &= c_{\boldsymbol{\xi}} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\imath(\boldsymbol{\xi}\cdot\boldsymbol{x})} e^{c_{\boldsymbol{\xi}} t} + \sum_{k=1}^{d} a_k \hat{u}_{0,\boldsymbol{\xi}} e^{c_{\boldsymbol{\xi}} t} 2^{i_k} \sum_{m=-r}^{r} \alpha_{k,i_k,m} e^{2\pi\imath(\boldsymbol{\xi}\cdot(\boldsymbol{x} + m 2^{-i_k} \boldsymbol{e}_k))} . \end{aligned}$$

Dividing out the common factor $\hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\imath(\boldsymbol{\xi}\cdot\boldsymbol{x})} e^{c_{\boldsymbol{\xi}} t}$ and re-arranging we obtain (3.20). Finally, for any given $t \geq 0$ the convergence of the Fourier series of $\omega(\boldsymbol{x}, t)$ when $\Re(c_{\boldsymbol{\xi}}) \leq 0$ follows from the convergence of the series for $u_0(\boldsymbol{x})$, in particular $\|\omega\|_2^2 = \sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} |\hat{u}_{0,\boldsymbol{\xi}} e^{c_{\boldsymbol{\xi}} t}|^2 \leq \sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} |\hat{u}_{0,\boldsymbol{\xi}}|^2 = \|u\|_2^2$ via Parseval's identity. □

Notice that $c_{\boldsymbol{\xi}}$ is simply a sum of the eigenvalues one obtains for the one dimensional advection problem corresponding to each of the dimensions $k = 1, \dots, d$. In particular we may write $c_{\boldsymbol{\xi}} = c_{\xi_1} + \cdots + c_{\xi_d}$ where each

$$c_{\xi_k} := -a_k 2^{i_k} \sum_{m=-r}^{r} \alpha_{k,i_k,m} e^{2\pi\imath m 2^{-i_k} \xi_k}$$

depends only on the ξ_k component of $\boldsymbol{\xi}$ (and only the kth component of $\underline{i}$ and $\boldsymbol{a}$). Further, for each k, as $i_k \to \infty$ one has $c_{\xi_k} \to -2\pi\imath\xi_k a_k$ as a consequence of consistency of the discretisation. This is evident in the following lemma.

Proposition 3.17. *Let* D_{i_k} *be a finite difference discretisation as in* (3.17) *which is a pth order consistent approximation of* $\frac{\partial}{\partial x_k}$ *with eigenvalues* c_{ξ_k} *such that* $D_{i_k} e^{2\pi\imath\xi_k x_k} = c_{\xi_k} e^{2\pi\imath\xi_k x_k}$*, then*

$$-a_k D_{i_k} e^{2\pi\imath\xi_k x_k} = \left(-2\pi\imath\xi_k a_k - \xi_k^{p+1} 2^{-p i_k} \eta_k\right) e^{2\pi\imath\xi_k x_k} , \tag{3.21}$$

where η_k *is uniformly bounded with respect to* ξ_k*.*

Proof. As a consequence of Lemma 3.16 we know that

$$c_{\xi_k} = -a_k 2^{i_k} \sum_{m=-r}^{r} \alpha_{k,i_k,m} e^{2\pi\iota m 2^{-i_k}\xi_k} .$$

Letting $h = 2^{-i_k}$ we use a Taylor expansion of $e^{2\pi\iota m h\xi_k}$ up to pth order around $h = 0$ we have

$$\begin{aligned} c_{\xi_k} &= -a_k h^{-1} \sum_{m=-r}^{r} \alpha_{k,i_k,m} \left(\sum_{q=0}^{p} \frac{(2\pi\iota m h\xi_k)^q}{q!} \right. \\ &\qquad\qquad \left. + (2\pi\iota m h\xi_k)^{p+1} \int_0^1 \frac{(1-t)^p}{p!} e^{2\pi\iota m h\xi_k t}\, dt \right) \\ &= -a_k h^{-1} \left(\sum_{q=0}^{p} \frac{(2\pi\iota h\xi_k)^q}{q!} \sum_{m=-r}^{r} \alpha_{k,i_k,m} m^q \right. \\ &\qquad \left. + (2\pi\iota h\xi_k)^{p+1} \sum_{m=-r}^{r} \alpha_{k,i_k,m} m^{p+1} \int_0^1 \frac{(1-t)^p}{p!} e^{2\pi\iota m h\xi_k t}\, dt \right) . \end{aligned}$$

Using the properties of the $\alpha_{k,m}$ from Lemma 3.15 one obtains

$$\begin{aligned} c_{\xi_k} &= -a_k h^{-1} 2\pi\iota h\xi_k - a_k h^{-1} (2\pi\iota h\xi_k)^{p+1} \sum_{m=-r}^{r} \alpha_{k,i_k,m} m^{p+1} \int_0^1 \frac{(1-t)^p}{p!} e^{2\pi\iota m h\xi_k t}\, dt \\ &= -2\pi\iota\xi_k a_k - h^p \xi_k^{p+1} \eta_k , \end{aligned}$$

where $\eta_k := a_k (2\pi\iota)^{p+1} \sum_{m=-r}^{r} \alpha_{k,i_k,m} m^{p+1} \int_0^1 \frac{(1-t)^p}{p!} e^{2\pi\iota m h\xi_k t}\, dt$. Substituting $h = 2^{-i_k}$ back in we obtain (3.21). Finally we notice that $|\eta_k| \le |a_k| \frac{(2\pi)^{p+1}}{(p+1)!} (2r+1) r^{p+1} \max_m |\alpha_{k,i_k,m}|$ which is independent of ξ_k. □

Our goal is to study the difference between the solutions ω of the discretised problem and the (exact) solutions u of the advection equation. For this purpose we define the error function

$$\epsilon(\boldsymbol{x}, t) := u(\boldsymbol{x}, t) - \omega(\boldsymbol{x}, t) . \tag{3.22}$$

We also define the operator $E_{\underline{i}} := \nabla - D_{\underline{i}}$, that is $E_{\underline{i}} = \left(\frac{\partial}{\partial x_1} - D_{i_1}, \ldots, \frac{\partial}{\partial x_d} - D_{i_d} \right)$, for which $\mathrm{E}_{\underline{i}} u$ gives the difference between the continuous and discrete spatial derivatives of u.

Proposition 3.18. *Fix $\underline{i} \in \mathbb{N}^d$ and let u and ω be solutions to* (3.16) *and* (3.19) *respectively, then $\epsilon(\boldsymbol{x}, t) = u(\boldsymbol{x}, t) - \omega(\boldsymbol{x}, t)$ is the unique solution to*

$$\frac{\partial \epsilon}{\partial t} = -\boldsymbol{a} \cdot D_{\underline{i}} \epsilon - \boldsymbol{a} \cdot E_{\underline{i}} u ,$$

with $\omega(\boldsymbol{x},t) \in L^2_{per}([0,1]^d)$ *and* $u(\boldsymbol{x},0) \in H^1_{per}([0,1]^d)$. *Further, one has*

$$\epsilon(\boldsymbol{x},t) = \exp(-t\boldsymbol{a}\cdot D_{\underline{i}})\epsilon(\boldsymbol{x},0) + u(\boldsymbol{x},t) - \exp(-t\boldsymbol{a}\cdot D_{\underline{i}})u(\boldsymbol{x},0)\,. \tag{3.23}$$

Before proceeding with the proof we note that $\exp(-t\boldsymbol{a}\cdot D_{\underline{i}}) := \sum_{n=0}^{\infty} \frac{(-t\boldsymbol{a}\cdot D_{\underline{i}})^n}{n!}$ is well defined as $\boldsymbol{a}\cdot D_{\underline{i}}$ is bounded, in particular

$$\|\boldsymbol{a}\cdot D_{\underline{i}}u\|_2 \leq \left(\sum_{k=1}^{d} |a_k| 2^{i_k} \sum_{m=-r}^{r} |\alpha_{k,m}|\right) \|u\|_2$$

for all $u \in L^2_{\text{per}}([0,1]^d)$.

Proof. We have $\frac{\partial\epsilon}{\partial t} = \frac{\partial u}{\partial t} - \frac{\partial\omega}{\partial t}$ and substituting equations (3.16) and (3.19) into the right hand side yields the PDE

$$\begin{aligned}\frac{\partial\epsilon}{\partial t} = \frac{\partial u}{\partial t} - \frac{\partial\omega}{\partial t} &= -\boldsymbol{a}\cdot\nabla u + \boldsymbol{a}\cdot D_{\underline{i}}\omega_{\underline{i}} \\ &= -\boldsymbol{a}\cdot(D_{\underline{i}}u - D_{\underline{i}}\omega + (\nabla - D_{\underline{i}})u) = -\boldsymbol{a}\cdot D_{\underline{i}}\epsilon - \boldsymbol{a}\cdot E_{\underline{i}}u\,.\end{aligned}$$

The expression (3.23) is obtained via

$$\begin{aligned}\epsilon(\boldsymbol{x},t) &= u(\boldsymbol{x},t) - \omega(\boldsymbol{x},t) \\ &= u(\boldsymbol{x},t) - \exp(-t\boldsymbol{a}D_{\underline{i}})\omega(\boldsymbol{x},0) \\ &= u(\boldsymbol{x},t) + \exp(-t\boldsymbol{a}D_{\underline{i}})\left(u(\boldsymbol{x},0) - \omega(\boldsymbol{x},0)\right) - \exp(-t\boldsymbol{a}D_{\underline{i}})u(\boldsymbol{x},0) \\ &= \exp(-t\boldsymbol{a}D_{\underline{i}})\epsilon(\boldsymbol{x},0) + u(\boldsymbol{x},t) - \exp(-t\boldsymbol{a}D_{\underline{i}})u(\boldsymbol{x},0)\,,\end{aligned}$$

as required. □

Note that Lastdrager et.al. also considered problems where $\boldsymbol{a}$ may vary over time in which case one must replace $-t\boldsymbol{a}\cdot D_{\underline{i}}$ in (3.23) with $-\int_0^t \boldsymbol{a}(\tau)\cdot D_{\underline{i}}\,d\tau$. We do not consider this case in detail. Further, we typically assume that the initial condition is exact, that is $\epsilon_{\underline{i}}(\boldsymbol{x},0) = 0$. We provide the proof for an identity which will be useful.

Lemma 3.19. *Let* $y_1,\dots,y_d \in \mathbb{R}$, *then*

$$y_1\cdots y_d - 1 = \sum_{k=1}^{d} \sum_{\substack{\{s_1,\dots,s_k\}\\ \subset\{1,\dots,d\}}} (y_{s_1}-1)\cdots(y_{s_k}-1)\,.$$

Proof. The case $d = 1$ is trivial, similarly for $d = 2$ one obtains via expansion

$$(y_1 - 1)(y_2 - 1) + (y_1 - 1) + (y_2 - 1) = y_1 y_2 - 1\,.$$

By induction, given the case 2 and $d - 1$ one has

$$\begin{aligned} y_1 \cdots y_d - 1 &= (y_1 \cdots y_{d-1})y_d - 1 \\ &= (y_1 \cdots y_{d-1} - 1)(y_d - 1) + (y_1 \cdots y_{d-1} - 1) + (y_d - 1) \\ &= \sum_{k=1}^{d-1} \sum_{\substack{\{s_1,\dots,s_k\} \\ \subset\{1,\dots,d\}}} (y_{s_1} - 1)\cdots(y_{s_k} - 1)(y_d - 1) \\ &\quad + \sum_{k=1}^{d-1} \sum_{\substack{\{s_1,\dots,s_k\} \\ \subset\{1,\dots,d\}}} (y_{s_1} - 1)\cdots(y_{s_k} - 1) + (y_d - 1)\,. \end{aligned}$$

For the sums in the last equality we observe for a fixed $k \in \{2, \dots, d-1\}$ that the first sum contributes $\binom{d-1}{k-1}$ terms which is a product of k terms of the form $(y_{s_l} - 1)$ (including $(y_d - 1)$) whilst the second sum contributes $\binom{d-1}{k}$ such terms. As these are all distinct then they must be all $\binom{d-1}{k-1} + \binom{d-1}{k} = \binom{d}{k}$ terms in the sum $\sum_{\substack{\{s_1,\dots,s_k\} \\ \subset\{1,\dots,d\}}} (y_{s_1} - 1)\cdots(y_{s_k} - 1)$. For the case $k = d$ then one obtains $(y_1 - 1)\cdots(y_d - 1)$ from the first of the two sums. Similarly, for the case $k = 1$ the last sum provides $(y_1 - 1) + \cdots + (y_{d-1} - 1)$ to which we add the $(y_d - 1)$. Therefore the identity holds for the case d and thus for all integers $d \geq 1$ by induction. $\square$

Before proceeding with the main result we first define a special class of functions whose mixed derivatives have a Fourier series which is absolutely convergent.

Definition 3.20. Given a function $u \in L^2(\Omega)$ with $\Omega \subset \mathbb{R}^d$ we define

$$\|u\|_A = \sum_{\boldsymbol{\xi} \in \mathbb{Z}^d} |\hat{u}_{\boldsymbol{\xi}}|\,,$$

where the $\hat{u}_{\boldsymbol{\xi}}$ are the Fourier coefficients of u. We define $H^p_{\text{mpa}}([0,1]^d)$ (mpa short for mixed, periodic, absolute) to be the functions in H^p_{mix} which are periodic and have a Fourier series which is absolutely convergent for all mixed derivatives, specifically

$$H^p_{\text{mpa}}([0,1]^d) = \left\{ u \in C(\mathbb{R}^d) : \begin{array}{c} u|_{[0,1]^d} \in H^p_{\text{mix}}([0,1]^d) \\ u(\boldsymbol{x}) = u(\boldsymbol{x} + \boldsymbol{e}_k) \quad \forall k \in \{1, \dots, d\} \\ \|D^{\underline{j}} u\|_A < \infty \quad \forall \underline{0} \leq \underline{j} \leq \underline{p} \end{array} \right\},$$

(with $\boldsymbol{e}_k$ the unit vector in the kth direction).

The reason we require the Fourier series to be absolutely convergent is that a particular finite difference discretisation may generate dispersion which causes the peaks of all of the basis functions to line up at some time during the evolution.

Theorem 3.21. *Let $\epsilon(\boldsymbol{x},t)$ be as in Proposition 3.18 with $\epsilon(\boldsymbol{x},0)=0$ and $u_0(\boldsymbol{x}) \in H^{p+1}_{mpa}([0,1]^d)$. Additionally, let each D_{i_k} be a pth order consistent approximation of $\frac{\partial}{\partial x_k}$ whose eigenvalues have non-positive real part. Then, $\epsilon(\boldsymbol{x},t)$ has the form of the error splitting*

$$\epsilon(\boldsymbol{x},t) = \sum_{k=1}^{d} \sum_{\substack{\{s_1,\dots,s_k\}\\ \subset\{1,\dots,d\}}} t^k 2^{-p(i_{s_1}+\dots+i_{s_k})} \gamma_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}}) \,, \tag{3.24}$$

and, further, there exists $K>0$ such that $|\gamma_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}})| < K$ for all $\{s_1,\dots,s_k\} \subset \{1,\dots,d\}$, $k=1,\dots,d$.

Proof. As $\epsilon(\boldsymbol{x},0)=0$ one has $\epsilon(\boldsymbol{x},t) = u(\boldsymbol{x},t) - \exp(-t\boldsymbol{a}\cdot D_{\underline{i}})u(\boldsymbol{x},0)$ from Proposition 3.18. Now given the Fourier series for $u(\boldsymbol{x},0)$ one has

$$\begin{aligned}
\epsilon(\boldsymbol{x},t) &= \left(\sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota\boldsymbol{\xi}\cdot(\boldsymbol{x}-\boldsymbol{a}t)} \right) - \exp(-t\boldsymbol{a}\cdot D_{\underline{i}}) \left(\sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota\boldsymbol{\xi}\cdot\boldsymbol{x}} \right) \\
&= \sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} \left(e^{-2\pi\iota\boldsymbol{\xi}\cdot\boldsymbol{a}t} - e^{c_{\boldsymbol{\xi}}t} \right) \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota\boldsymbol{\xi}\cdot\boldsymbol{x}} \\
&= \sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} \left(1 - e^{t(c_{\boldsymbol{\xi}}+2\pi\iota\boldsymbol{\xi}\cdot\boldsymbol{a})} \right) \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota\boldsymbol{\xi}\cdot(\boldsymbol{x}-\boldsymbol{a}t)} \,.
\end{aligned}$$

Defining $\epsilon_{\boldsymbol{\xi}}(t) := 1 - e^{2\pi\iota\boldsymbol{\xi}\cdot\boldsymbol{a}t + c_{\boldsymbol{\xi}}t}$ we may write

$$\epsilon(\boldsymbol{x},t) = \sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} \epsilon_{\boldsymbol{\xi}}(t) \hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota\boldsymbol{\xi}\cdot(\boldsymbol{x}-\boldsymbol{a}t)} \,.$$

Now as $c_{\boldsymbol{\xi}} = c_{\xi_1} + \dots + c_{\xi_d}$ with each c_{ξ_k} depending only on the ξ_k component of $\boldsymbol{\xi}$ (and only the kth component of $\underline{i}$ and $\boldsymbol{a}$) we can write this as

$$\begin{aligned}
\epsilon_{\boldsymbol{\xi}}(t) &= 1 - e^{t\sum_{k=1}^{d} 2\pi\iota\xi_k a_k + c_{\xi_k}} \\
&= 1 - \prod_{k=1}^{d} e^{t(2\pi\iota\xi_k a_k + c_{\xi_k})} \,.
\end{aligned}$$

Further, by rewriting the identity of Lemma 3.19 as

$$1 - y_1\cdots y_d = \sum_{k=1}^{d} (-1)^{k-1} \sum_{\substack{\{s_1,\dots,s_k\}\\ \subset\{1,\dots,d\}}} (1-y_{s_1})\cdots(1-y_{s_k})$$

we can decompose $\epsilon_{\boldsymbol{\xi}}(t)$ as

$$\epsilon_{\boldsymbol{\xi}}(t) = \sum_{k=1}^{d}(-1)^{k-1} \sum_{\substack{\{s_1,\dots,s_k\} \\ \subset\{1,\dots,d\}}} \prod_{l=1}^{k} \epsilon_{\xi_{s_l}}(t)\,, \tag{3.25}$$

where

$$\epsilon_{\xi_{s_l}}(t) = \left(1 - e^{t(2\pi\iota\xi_{s_l}a_{s_l}+c_{\xi_{s_l}})}\right)\,.$$

Now as a consequence of Proposition 3.17 we have that

$$c_{\xi_k} = -2\pi\iota\xi_k a_k - h_{i_k}^p \xi_k^{p+1}\eta_k$$

and thus with $k = s_l$

$$\epsilon_{\xi_{s_l}}(t) = \left(1 - e^{-th_{i_{s_l}}^p \xi_{s_l}^{p+1}\eta_{s_l}}\right)\,.$$

Now we perform a Taylor series expansion of $e^{-th_{i_k}^p \xi_k^{p+1}\eta_k}$. Note that as η_k is complex we must take care to determine the remainder via complex contour integrals. As the eigenvalues c_{ξ_k} of D_{i_k} have non-positive real part then it follows that $-th_{i_k}^p \xi_k^{p+1}\eta_k$ must have non-positive real part. Consider the function e^z with $z \in \mathbb{C}$, then as e^z is analytic everywhere then we have via Cauchy's integral formula

$$\begin{aligned} e^z &= \sum_{n=0}^{\infty} \frac{z^n}{2\pi\iota}\int_{\mathcal{C}} \frac{e^w}{w^{n+1}}\,dw \\ &= 1 + \frac{1}{2\pi\iota}\int_{\mathcal{C}} \frac{e^w}{w}\sum_{n=1}^{\infty}\left(\frac{z}{w}\right)^n dw = 1 + \frac{z}{2\pi\iota}\int_{\mathcal{C}} \frac{e^w}{w(w-z)}\,dw\,, \end{aligned}$$

where $\mathcal{C}$ is some closed path containing the origin and z. Now letting $\tilde{\eta} = \frac{1}{2\pi\iota}\int_{\mathcal{C}} \frac{e^w}{w(w-z)}\,dw$ we have $e^z = 1 + z\tilde{\eta}$. It follows that with $\tilde{\eta}_{s_l} = \tilde{\eta}$ one has

$$\epsilon_{\xi_{s_l}}(t) = 1 - e^{-th_{i_{s_l}}^p \xi_{s_l}^{p+1}\eta_{s_l}} = th_{i_{s_l}}^p \xi_{s_l}^{p+1}\eta_{s_l}\tilde{\eta}_{s_l}\,. \tag{3.26}$$

Now as each c_{ξ_k} has non-positive real part it follows that $|\epsilon_{\xi_{s_l}}(t)| \le 2$ and thus $|\tilde{\eta}_{s_l}| \le \frac{2}{th_{i_{s_l}}^p|\eta_{s_l}|}$ independent of $\xi_{s_l} \ge 1$ (and $\tilde{\eta}_{s_l} = 0$ if $\xi_{s_l} = 0$). Substituting (3.26) into (3.25) we obtain

$$\epsilon_{\boldsymbol{\xi}}(t) = \sum_{k=1}^{d}(-1)^{k-1} \sum_{\substack{\{s_1,\dots,s_k\} \\ \subset\{1,\dots,d\}}} t^k 2^{-p(i_{s_1}+\cdots+i_{s_k})}(\xi_{s_1}\cdots\xi_{s_k})^{p+1}\gamma_{s_1}\cdots\gamma_{s_k}\,,$$

where $\gamma_{s_k} := \eta_{s_k}\tilde{\eta}_{s_k}$. As η_{s_k} and $\tilde{\eta}_{s_k}$ are bounded independently of ξ_{s_k} then γ_{s_k} is also bounded independently of ξ_{s_k}. Thus we have an error splitting formula for the error of the eigenmode $e^{2\pi\iota\boldsymbol{\xi}\cdot\boldsymbol{x}}$. In the sum over all eigenmodes we have

$$\begin{aligned}\epsilon(\boldsymbol{x},t) &= \sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} \epsilon_{\boldsymbol{\xi}}(t)\hat{u}_{0,\boldsymbol{\xi}} e^{2\pi\iota\boldsymbol{\xi}\cdot(\boldsymbol{x}-\boldsymbol{a}t)} \\ &= \sum_{k=1}^{d} \sum_{\substack{\{s_1,\dots,s_k\}\\ \subset\{1,\dots,d\}}} t^k 2^{-p(i_{s_1}+\cdots+i_{s_k})}\gamma_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}})\,,\end{aligned}$$

where

$$\gamma_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}}) := (-1)^{k-1}\sum_{\boldsymbol{\xi}\in\mathbb{Z}^d} \hat{u}_{0,\boldsymbol{\xi}}(\xi_{s_1}\cdots\xi_{s_k})^{p+1}(\gamma_{s_1}\cdots\gamma_{s_k})e^{2\pi\iota\boldsymbol{\xi}\cdot(\boldsymbol{x}-\boldsymbol{a}t)}\,.$$

Further, using Holder's inequality we have that

$$\begin{aligned}|\gamma_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}})| &\le \left(\sup_{\boldsymbol{\xi}}|\gamma_{s_1}\cdots\gamma_{s_k}|\right)\sum_{\boldsymbol{\xi}\in\mathbb{Z}^d}|\hat{u}_{0,\boldsymbol{\xi}}\xi_{s_1}\cdots\xi_{s_k}|^{p+1} \\ &= \left(\sup_{\boldsymbol{\xi}}|\gamma_{s_1}\cdots\gamma_{s_k}|\right)\left\|\frac{\partial^{k(p+1)}u_0}{\partial x_{s_1}^{p+1}\cdots\partial x_{s_k}^{p+1}}\right\|_A\,,\end{aligned}$$

where the right hand side is finite as each of the γ_{s_l} is uniformly bounded and $\|D^{\underline{j}}u_0(\boldsymbol{x})\|_A < \infty$ for all $\underline{0} \le \underline{j} \le \underline{p+1}$. Thus by taking $K > 0$ to be the maximum of the $\gamma_{s_1,\dots,s_k}(2^{i_{s_1}},\dots,2^{i_{s_k}})$ for all $k = 1,\dots,d$ and $\{s_1,\dots,s_k\} \subset \{1,\dots,d\}$ then $\epsilon(\boldsymbol{x},t)$ satisfies the desired error splitting model. $\square$

For a fixed t, by setting $K_t = \max\{K, t^dK\}$, an error bound for the combination of technique applied to the $\omega_{\underline{i}}$ follows immediately from Theorem 2.27, that is if we use u_n^c to denote the level n combination of the $u_{\underline{i}}$ (which is each the solution of $\frac{\partial u}{\partial t} + \boldsymbol{a}\cdot D_{\underline{i}}u = 0$) then

$$|u - u_n^c| \le K_t 2^{-pn}(1+2^p)^{d-1}\binom{n+2d-1}{d-1}\,.$$

Lastdrager et.al. considered truncated combinations whereby one discards grids from the combination technique which do not have a specified minimum level of discretisation in each dimension. We consider such combinations in greater detail in Section 4.1. As we have shown the spatial discretisation satisfies the error splitting model many of the results in developed in Section 4.1 will be directly applicable to the advection problem solved with an explicit finite difference method.

In addition to the bound obtained via Theorem 2.27 we will also study what happens to each of the error terms $t^k\gamma_{l_1,\dots,l_k}(\boldsymbol{x},t)h_{i_{l_1}}^p \cdots h_{i_{l_k}}^p$ individually to better understand the effect of repeated combinations on the solution error. We can apply Lemma 2.32 to each of these terms to find that the level n combination technique applied to $t^k\gamma_{l_1,\dots,l_k}(\boldsymbol{x},t)h_{i_{l_1}}^p \cdots h_{i_{l_k}}^p$ yields

$$t^k 2^{-pn} \sum_{m=0}^{k-1} \binom{k-1}{m} (-2^p)^m \left(\sum_{i_{l_1}+\cdots+i_{l_k}=n-m} \gamma_{l_1,\dots,l_k}(\boldsymbol{x},t) \right) . \tag{3.27}$$

Let $K > 0$ be such that $|\gamma_{l_1,\dots,l_k}(\boldsymbol{x},t)| \leq K$ for all $\boldsymbol{x}$, $\underline{i}$, $k \in \{1,\dots,d\}$ and $\{l_1,\dots,l_k\} \subset \{1,\dots,d\}$, then the absolute value of (3.27) is bounded by

$$\begin{aligned} K_t' t^k 2^{-pn} \sum_{m=0}^{k-1} \binom{k-1}{m} |-2^p|^m \binom{n-m+k-1}{k-1} & \\ \leq K_t' t^k 2^{-pn} \binom{n+k-1}{k-1} (1+2^p)^{k-1} , \end{aligned}$$

Which leads to a result for $|u-u_n^c|$ analogous to (2.26) in the proof of Theorem 2.27, namely

$$|u-u_n^c| \leq K 2^{-pn} \sum_{k=1}^{d} t^k \binom{d}{k} \binom{n+k-1}{k-1} (1+2^p)^{k-1} . \tag{3.28}$$

Suppose now that the solutions on the coarse grids were to be combined at time t/M where M is a positive integer. One has that the leading error terms (3.27) has the additional factor M^{-k}. During the next evolution of t/M these leading error terms from the previous evolution become the initial error, that is from (3.23) we consider

$$\hat{\epsilon}_{\underline{i}}(2t/M) = e^{-(t/M)\boldsymbol{a}\cdot D_{\underline{i}}} \epsilon(\boldsymbol{x},t/M) + u(\boldsymbol{x},2t/M) - e^{-(t/M)\boldsymbol{a}\cdot D_{\underline{i}}} u(\boldsymbol{x},t/M)$$

Now the $u(\boldsymbol{x},2t/M) - e^{-(t/M)\boldsymbol{a}\cdot D_{\underline{i}}} u(\boldsymbol{x},t/M)$ term satisfies the same bound as $\epsilon_{\underline{i}}(\boldsymbol{x}-\boldsymbol{a}t/M,t/M)$ as the only difference is the translation of u by $\boldsymbol{a}t/M$. On the other hand the $e^{-(t/M)\boldsymbol{a}\cdot D_{\underline{i}}}\epsilon(\boldsymbol{x},t/M)$ term is evolution of the error generated via the discrete advection operator. If the discrete operator were exact this would just be a translation. Of course this is not the case and there is an additional error term generated. However, as the $\epsilon_{\underline{i}}(\boldsymbol{x}-\boldsymbol{a}t/M,t/M)$ term is $\mathcal{O}\left(\frac{t}{M}2^{-pn}n^{d-1}\right)$ and the discrete advection operator is order p it is reasonable to assume that this contribution to the error is negligible compared to the other terms. Thus at time

$2t/M$ with combinations at time t/M and $2t/M$ the error is bounded above by

$$K_t'2^{-pn}\sum_{k=1}^{d}(2+\delta)\left(\frac{t}{M}\right)^k\binom{d}{k}\binom{n+k-1}{k-1}(1+2^p)^{k-1},$$

where δ bounds the contribution of $e^{-(t/M)\boldsymbol{a}\cdot D_{\underline{i}}}\epsilon(\boldsymbol{x},t/M)-\epsilon(\boldsymbol{x}-\boldsymbol{a}t/M,t/M)$ for which we assume $|\delta|\ll 1$. We may now repeat this argument for additional combinations at times $\frac{3t}{M},\frac{4t}{M},\dots,t$. For the last combination at time t one obtains the bound

$$K_t'2^{-pn}\sum_{k=1}^{d}(M+\delta')\left(\frac{t}{M}\right)^k\binom{d}{k}\binom{n+k-1}{k-1}(1+2^p)^{k-1},$$

where the δ' term encapsulates the error of the discrete advection operator applied to the $\hat{\epsilon}_{\underline{i}}$ from previous steps which we assume satisfies $|\delta'|\ll 1$. Thus we note that for the $k=1$ in the sum we have $(M+\delta')\frac{t}{M}\approx t$ and thus repeated combinations has no impact on this term. However, for the $k>1$ in the sum we have $(M+\delta')\left(\frac{t}{M}\right)^k\approx t^kM^{-k+1}$ which leads to a reduction in the error. In particular, the $k=d$ in the sum (which contributes the large term $t^k\binom{n+d-1}{d-1}(1+2^p)^{d-1}$ to the error when there is only one combination at time $t>1$) has a reduction of M^{-d+1}. Whether this is noticed in practice depends on how tight a bound K_t' is to the γ terms corresponding to the $k=d$ term in the sum. In the limit $M\to\infty$ the $k>1$ terms vanish leaving $K_t'2^{-pn}td$. However, this ignores the possibility of terms accumulating in the δ' which may become a significant contribution to the error for large M.

Remark 3.22. Theorem 3.21 shows that MOL solutions of advection from a specific family of finite difference discretisations satisfy an error splitting when the finite difference operator is applied over the entire domain (as opposed to grid points) and the time integration is exact. Of course, in practice this is not how we compute solutions to advection. However, the result is still relevant as we shall point out here.

- If one samples the MOL solution $\omega_{\underline{i}}$ on a grid $\Omega_{\underline{i}}$ at time t, that is $\omega_{\underline{i}}(\Omega_{\underline{i}},t)$, then the result is the same as solving the ODE system

$$\frac{\partial\omega(\Omega_{\underline{i}},t)}{\partial t}+\boldsymbol{a}\cdot D_{\underline{i}}\omega(\Omega_{\underline{i}},t)=0,$$

with initial condition $\omega(\Omega_{\underline{i}},0)=u_0(\Omega_{\underline{i}})$, up to time t. Clearly the resulting $\omega_{\underline{i}}(\Omega_{\underline{i}},t)$ satisfy the error splitting as described by the theorem at each grid point.

- In practice one computes the combination technique approximation by interpolating the function values on $\Omega_{\underline{i}}$ to the sparse grid (or a full grid containing the sparse grid). This is either done explicitly or implicitly via the hierarchisation approach. It is reasonable to expect that interpolation/quadrature methods based on tensor product rules also satisfy an error splitting. For example, Reisinger shows [110] that for $v \in H^2_{0,\text{mix}}$ then the piecewise multi-linear interpolant function samples on the grid $\Omega_{\underline{i}}$ (denoted $\mathcal{I}_{\underline{i}}u$) satisfies the pointwise error splitting

$$v - \mathcal{I}_{\underline{i}}v = \sum_{k=1}^{d} \sum_{\substack{\{s_1,\dots,s_k\} \\ \subseteq\{1,\dots,d\}}} \alpha_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}})4^{-i_{s_1}}\cdots 4^{-i_{s_k}} \,,$$

  and there exists $L > 0$ such that each $|\alpha_{s_1,\dots,s_k}| \leq L$ for all $\underline{i}$. More generally if $v \in H^q_{0,\text{mix}}$ we assume we have an interpolation method $\mathcal{I}'_{\underline{i}}$ which is order q and satisfies

$$v - \mathcal{I}'_{\underline{i}}v = \sum_{k=1}^{d} \sum_{\substack{\{s_1,\dots,s_k\} \\ \subseteq\{1,\dots,d\}}} \alpha_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}})2^{-qi_{s_1}}\cdots 2^{-qi_{s_k}} \,.$$

  If $q \geq p$ then applying such an integration method to the $\omega_{\underline{i}}(t)$ one has

$$u(t) - \mathcal{I}'_{\underline{i}}\omega_{\underline{i}}(t) = \sum_{k=1}^{d} \sum_{\substack{\{s_1,\dots,s_k\} \\ \subset\{1,\dots,d\}}} t^k 2^{-p(i_{s_1}+\cdots+i_{s_k})}\gamma'_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}}) \,,$$

  (with $u(t)$ being the exact solution of the advection equation at time t) as the error splitting terms from the interpolation can be effectively absorbed into the original $\gamma_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}})$ to form $\gamma'_{s_1,\dots,s_k}(2^{-i_{s_1}},\dots,2^{-i_{s_k}})$.

- Lastly, we must consider that our ODE is not integrated exactly in practice. However, we assume that the ODE solver used is of sufficiently high order that the spatial errors are dominant for all stable choices of Δt. Under this assumption it is clear that as $\Delta t, \Delta x_1, \dots, \Delta x_d \to 0$ (with Δt always such that the scheme is stable) convergence of the combination solutions will follow from analysis of the spatial errors. In our numerical results in Sections 4.5 and 5.2.4 we typically use the classical fourth order Runge–Kutta method to solve the ODE obtained via second order discretisation of spatial derivatives.

Chapter 4

Variations and Generalisations of the Combination Technique

Many variations and generalisations of the combination technique have been developed in the literature. In Section 4.1 we will discuss the truncated combination technique, a variation which avoids some of the strongly anisotropic grids of the classical combination technique. Error estimates from Chapter 2 will be adapted to truncated combinations. In Section 4.2 we consider dimension adaptive sparse grids which inspired much of the initial work on fault tolerant adaptations of the combination technique. In particular, we review the framework of projections onto function space lattices developed in [70] and extend this via a study of projections onto hierarchical surpluses. Section 4.3 considers the use of multi-variate extrapolation within the combination technique to obtain high order solutions. Combined with the adaptive sparse grids framework one obtains an adaptive extrapolation algorithm. Section 4.4 develops a generalisation of the combination technique for combining arbitrary collections of grids. Two approaches for the computation of coefficients will be considered based upon the work in Section 4.2. Much of the work in the latter three sections is featured in the publications [69, 65, 64].

4.1 Truncated Combination Technique

The classical combination technique

$$u_n^c := \sum_{k=0}^{d-1} (-1)^k \binom{d-1}{k} \sum_{|\underline{i}|=n-k} u_{\underline{i}}$$

is typically not used 'as is' in practice. There are often problems with using grids where the refinement is extremely coarse in all but one dimension which is extremely fine. For example the grid corresponding to the multi-index $(n, 0, \dots, 0)$ within the set $\{\underline{i} \in \mathbb{N}^d : |\underline{i}| = n\}$. Such grids are referred to as strongly anisotropic. There are several reasons why the inclusion of these grids can lead to poor results. For example, the initial condition may be poorly represented on such grids or the numerical scheme may be ill-suited to the elongated cells. In theory some of the error on such grids should cancel with that on nearby grids (that is grids corresponding to the multi-index $\underline{j}$ with $|\underline{j} - \underline{i}|$ small) but, in practice, what remains can be significant. A solution to this is to simply omit some of the strongly anisotropic grids. This is equivalent to truncating the sums used in the classical combination, hence leading to the so called truncated combination technique [9]. Leentvaar [86] extends the classical analysis based on the point-wise error splitting to these truncated combinations but with the restriction that the corresponding full grid is isotropic. Here we provide results for truncated combinations for which the corresponding full grid may be anisotropic. There are two important reasons for doing this. First, the rate of convergence may differ in the different dimensions for some applications which means we may wish to have finer discretisation in the slowest converging dimensions. Second, we may have different requirements for the minimum discretisation in each dimension which also leads to an isotropic full grid.

There are several ways in which a truncated combination technique may be defined. Fundamentally a truncated combination is just a translation of a classical combination with respect to the multi-indices $\underline{i}$ corresponding to the coarse/component solutions $u_{\underline{i}}$. As such we will use the following definition.

Definition 4.1. Given $n \geq 0$ and $\underline{s} \in \mathbb{N}^d$ we define the truncated combination as

$$u^t_{n,\underline{s}} := \sum_{k=0}^{d-1} (-1)^k \binom{d-1}{k} \sum_{\{\underline{i} \in \mathbb{N}^d : |\underline{i}| = n-k\}} u_{\underline{i}+\underline{s}} \,. \tag{4.1}$$

The sum over $\{\underline{i} \in \mathbb{N}^d : |\underline{i}| = n - k\}$ will be typically abbreviated to $\sum_{|\underline{i}|=n-k}$ and is defined to be 0 when $n - k < 0$. Here n corresponds to the level of the classical combination for which we translate the multi-indices by $\underline{s}$. Equation (4.1) may be expressed as a classical combination of level $n + |\underline{s}|$ with the sum over $\underline{i}$

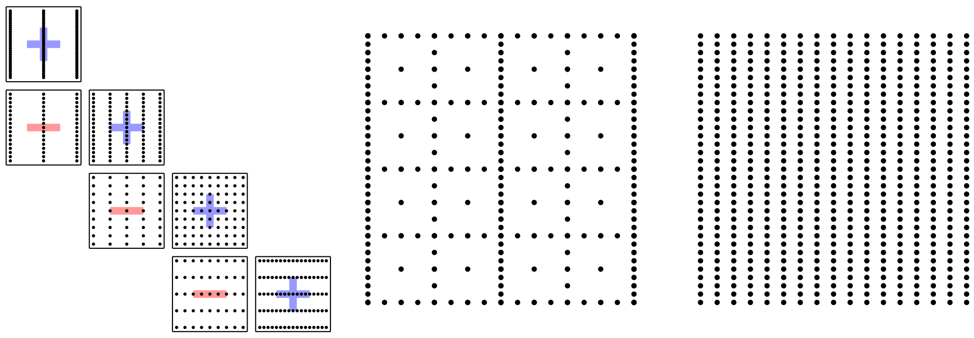


Figure 4.1: *From left to right we depict the two dimensional truncated combination $u^t_{3,(1,2)}$, the corresponding sparse grid points and the corresponding full grid points.*

truncated/restricted to $\underline{i} \geq \underline{s}$, that is

$$u^t_{n,\underline{s}} = \sum_{k=0}^{d-1} (-1)^k \binom{d-1}{k} \sum_{\substack{|\underline{i}|=n+|\underline{s}|-k \\ \underline{i} \geq \underline{s}}} u_{\underline{i}} \,.$$

Given a truncated combination $u^t_{n,\underline{s}}$ it will be useful to define the corresponding full grid approximation. The full grid approximation $u_{\underline{f}}$ corresponding to a truncated combination $u^t_{n,\underline{s}}$ is the approximation on the smallest grid which contains all of the grids that the $u_{\underline{i}+\underline{s}}$ where computed on. It is clear that $f_k = s_k + n$ for each k which we sometimes write as $\underline{f} = \underline{s} + \underline{n}$. An example of a truncated combination and the corresponding sparse and full grids is depicted in Figure 4.1.

Remark 4.2. We depart momentarily to comment on combinations which have been truncated relative to a full grid. Given a full grid index $\underline{f}$ and a level $n \leq |\underline{f}|$ one might have thought to consider the combination

$$\sum_{k=0}^{d-1} (-1)^k \binom{d-1}{k} \sum_{\substack{|\underline{i}|=n-k \\ \underline{i} \leq \underline{f}}} u_{\underline{i}} \,.$$

However there is a problem with this combination if $n > \min\{f_1, \dots, f_d\}$. For example, in 2 spatial dimensions the above reduces to

$$\sum_{\substack{i_1+i_2=n \\ (i_1,i_2)\leq(f_1,f_2)}} u_{i_1,i_2} - \sum_{\substack{i_1+i_2=n-1 \\ (i_1,i_2)\leq(f_1,f_2)}} u_{i_1,i_2}$$

for which the combination coefficients sum to -1 if $n > \min\{f_1, f_2\}$. It follows that this combination is not consistent and does not approximate the desired solution. This can be corrected by modifying the coefficients for the $k > 0$ terms using the theory of adaptive sparse grids which is developed in Section 4.2. In $d = 2$ dimensions one obtains the truncated combination $u^t_{f_1+f_2-n,(f_1-n,f_2-n)}$ but for $d \geq 3$ the resulting combination is typically no longer a truncated combination as we have defined. These are unusual but interesting combinations which will not be developed further in this section. We note that the results of Section 4.2 can be applied to such combinations.

As was done for classical combinations, we will consider the number of unknowns and the approximation error for truncated combinations. Given the truncated combination $u^t_{n,\underline{s}}$, the number of unknowns in each grid $u_{\underline{i}+\underline{s}}$, denoted as $\#(u_{\underline{i}+\underline{s}})$, satisfies

$$\#(u_{\underline{i}+\underline{s}}) = \prod_{k=1}^{d}(2^{i_k+s_k}+1) \leq \prod_{k=1}^{d} 2^{s_k}(2^{i_k}+1) = 2^{|\underline{s}|}\prod_{k=1}^{d}(2^{i_k}+1)\,.$$

It follows that the total number of unknowns required for the computation of $u^t_{n,\underline{s}}$ is bounded above by $2^{|\underline{s}|}$ times the number of unknowns required for the computation of u^c_n (see Section 2.2).

We now consider the approximation error of $u^t_{n,\underline{s}}$ when each of the $u_{\underline{i}+\underline{s}}$ is the usual piecewise linear interpolants. A rough bound is obtained via the following proposition.

Proposition 4.3. *Let $u \in H^2_{0,mix}$, $n \geq 0$, $\underline{s} \in \mathbb{N}^d$ and $u_{\underline{i}}$ be the usual piecewise linear interpolant of u for all $\underline{i} \in \mathbb{N}^d$. Further, let $\underline{f} = \underline{s} + \underline{n}$ (with $\underline{n} = (n, \ldots, n)$), and*

$$\epsilon^c_n := 2^{-2n}\frac{1}{3}\left(\frac{1}{3}\right)^d \|D^{\underline{2}}u\|_2 \sum_{k=0}^{d-1}\binom{n}{k}\left(\frac{1}{3}\right)^{d-1-k}\,.$$

The truncated combination technique $u^t_{n,\underline{s}}$ satisfies the error bound

$$\|u_{\underline{f}} - u^t_{n,\underline{s}}\|_2 \leq \epsilon^c_{n+|\underline{s}|}\,.$$

Recall that when the $u_{\underline{i}}$ are piecewise linear interpolants then one has $u^s_n = u^c_n$ and therefore from Proposition 2.19 one has $\|u - u^c_n\|_2 = \|u - u^s_n\|_2 \leq \epsilon^c_n$. Thus this result says we can bound the difference between interpolation onto the full grid and the interpolant obtained via the truncated combination by a classical combination error bound for $u^c_{n+|\underline{s}|}$.

Proof. As $u^h_{\underline{i}}$ denotes the level $\underline{i}$ hierarchical surplus of u (see Section 2.1) we may write $u_{\underline{f}} = \sum_{\underline{i} \leq \underline{f}} u^h_{\underline{i}}$ and as $u^t_{n,\underline{s}}$ contains all $u^h_{\underline{i}}$ with $|\underline{i}| \leq n + |\underline{s}|$ with the exception of $\underline{i} \not\leq \underline{s} + \underline{n}$ one may also write

$$u^t_{n,\underline{s}} = \sum_{\substack{|\underline{i}| \leq n + |\underline{s}| \\ \underline{i} \leq \underline{s} + \underline{n}}} u^h_{\underline{i}} \,.$$

As $\underline{f} = \underline{s} + \underline{n}$ it follows that

$$\begin{aligned} \|u_{\underline{f}} - u^t_{n,\underline{s}}\|_2 &= \left\| \sum_{\{\underline{i} \in \mathbb{N}^d : |\underline{i}| > n + |\underline{s}| \text{ and } \underline{i} \leq \underline{s} + \underline{n}\}} u^h_{\underline{i}} \right\|_2 \\ &\leq \sum_{\{\underline{i} \in \mathbb{N}^d : |\underline{i}| > n + |\underline{s}| \text{ and } \underline{i} \leq \underline{s} + \underline{n}\}} \|u^h_{\underline{i}}\|_2 \\ &\leq \sum_{|\underline{i}| > n + |\underline{s}|} \|u^h_{\underline{i}}\|_2 \\ &\leq \epsilon^c_{n + |\underline{s}|} \,, \end{aligned}$$

where the last inequality is obtained via the proof of Proposition 2.19. □

Corollary 4.4. *Let u, $u_{\underline{i}}$, $n \geq 0$, $\underline{s} \in \mathbb{N}^d$ and ϵ^c_n be as in Proposition 4.3. Let $\underline{f} = \underline{s} + \underline{n}$ and*

$$\epsilon_{\underline{f}} := 9^{-d} \|D^2 u\|_2 \sum_{k=1}^{d} 4^{-f_k} \,,$$

then

$$\|u - u^t_{n,\underline{s}}\|_2 \leq \epsilon_{\underline{f}} + \epsilon^c_{n + |\underline{s}|} \,.$$

Proof. This follows from the triangle inequality as

$$\|u - u^t_{n,\underline{s}}\|_2 = \|u - u_{\underline{f}} + u_{\underline{f}} - u^t_{n,\underline{s}}\|_2 \leq \|u - u_{\underline{f}}\|_2 + \|u_{\underline{f}} - u^t_{n,\underline{s}}\|_2$$

and $\|u - u_{\underline{f}}\|_2 \leq \epsilon_{\underline{f}}$ by Lemma 2.20. □

A similar result may also be obtained for the ∞ or energy norms by replacing ϵ^c_n and $\epsilon_{\underline{f}}$ with the appropriate bounds found in [22]. The result is given for the case where $u \in H^2_{0,\text{mix}}$ but it should be clear how this can be extended to $u \in H^2_{\text{mix}}$ similar to Proposition 2.22.

Proposition 4.3 gives us a simple way to estimate the interpolation error for truncated combinations (compared to the full grid interpolant) using the error estimate from the classical combination $u^c_{n+|\underline{s}|}$. However, the bound is generally an overestimate. For example, if $n = 0$ (and thus $\underline{f} = \underline{s}$) then $u_{\underline{f}} = u^t_{0,\underline{f}}$ and thus the difference is 0. Obtaining a tighter bound requires a bit more work, we provide a tighter bound for two and three dimensions to help illustrate how the truncated combination differs from a classical combination.

Proposition 4.5. *Let $n \geq 0$, $\underline{s} \in \mathbb{N}^d$, $\underline{f} = \underline{s} + \underline{n}$, $u \in H^2_{0,mix}([0,1]^d)$ and $u_{\underline{i}}$ be the usual piecewise linear interpolant of u for $\underline{i} \in \mathbb{N}^d$. Then for the case $d = 2$ one has*

$$\|u_{\underline{f}} - u^t_{n,\underline{s}}\|_2 \leq 3^{-4} \|D^{\underline{2}} u\|_2 2^{-2(n+|\underline{s}|)} \left(3n - 1 + 4^{-n}\right) .$$

Similarly, for the case $d = 3$ one has

$$\|u_{\underline{f}} - u^t_{n,\underline{s}}\|_2 \leq 3^{-6} \|D^{\underline{2}} u\|_2 2^{-2(n+|\underline{s}|)} \left(\frac{1}{2}(9n^2 + 51n - 22) + 12 \cdot 2^{-2n} - 2^{-4n}\right) .$$

Proof. As in Proposition 4.3 we observe that

$$u_{\underline{f}} - u^t_{n,\underline{s}} = \sum_{\substack{\underline{i} \leq \underline{f} \\ |\underline{i}| > n + |\underline{s}|}} u^h_{\underline{i}} .$$

From the triangle equality it follows that

$$\|u_{\underline{f}} - u^t_{n,\underline{s}}\|_2 \leq \sum_{|\underline{i}| > n + |\underline{s}| \,\&\, \underline{i} \leq \underline{f}} \|u^h_{\underline{i}}\|_2 = \sum_{|\underline{i}| > n \,\&\, \underline{i} \leq \underline{n}} \|u^h_{\underline{i}+\underline{s}}\|_2 .$$

We now need only estimate the sum in each case. For $d = 2$ one has

$$\begin{aligned}
\sum_{|\underline{i}| > n \,\&\, \underline{i} \leq \underline{n}} \|u^t_{\underline{i}+\underline{s}}\|_2 &= \sum_{k=n+1}^{2n} \sum_{i=k-n}^{n} \|u^h_{i+s_1, k-i+s_2}\|_2 \\
&\leq \sum_{k=n+1}^{2n} \sum_{i=k-n}^{n} 3^{-2} \|D^{\underline{2}} u\|_2 2^{-2(k+|\underline{s}|)} \\
&= 3^{-2} \|D^{\underline{2}} u\|_2 2^{-2|\underline{s}|} \sum_{k=n+1}^{2n} 2^{-2k} (2n - k + 1) \\
&= 3^{-4} \|D^{\underline{2}} u\|_2 2^{-2(|\underline{s}|+n)} \left(3n - 1 + 4^{-n}\right) .
\end{aligned}$$

Similarly for $d = 3$ one has

$$\begin{aligned}
\sum_{|\underline{i}| > n \,\&\, \underline{i} \leq \underline{n}} \|u^h_{\underline{i}+\underline{s}}\|_2 &= \sum_{k=n+1}^{3n} \sum_{|\underline{i}| = k \,\&\, \underline{i} \leq \underline{n}} \|u^h_{\underline{i}+\underline{s}}\|_2 \\
&= \sum_{k=n+1}^{2n-1} \sum_{|\underline{i}| = k \,\&\, \underline{i} \leq \underline{n}} \|u^h_{\underline{i}+\underline{s}}\|_2 + \sum_{k=2n}^{3n} \sum_{|\underline{i}| = k \,\&\, \underline{i} \leq \underline{n}} \|u^h_{\underline{i}+\underline{s}}\|_2 .
\end{aligned}$$

For the latter of the sums it is straightforward to show $|\{|\underline{i}| = k \,\&\, \underline{i} \leq \underline{n}\}| =$

$\binom{3n-k+2}{2}$ for $2n \le k \le 3n$ and therefore using the estimate of Lemma 2.15

$$\begin{aligned}\sum_{k=2n}^{3n} \sum_{|\underline{i}|=k \,\&\, \underline{i}\le\underline{n}} \|u^h_{\underline{i}+\underline{s}}\|_2 &\le \sum_{k=2n}^{3n} \sum_{|\underline{i}|=k \,\&\, \underline{i}\le\underline{n}} 3^{-3}\|D^{\underline{2}}u\|_2 2^{-2(k+|\underline{s}|)} \\ &= 3^{-3}\|D^{\underline{2}}u\|_2 2^{-2|\underline{s}|} \sum_{k=2n}^{3n} 2^{-2k} \sum_{|\underline{i}|=k \,\&\, \underline{i}\le\underline{n}} 1 \\ &= 3^{-3}\|D^{\underline{2}}u\|_2 2^{-2|\underline{s}|} \sum_{k=2n}^{3n} 2^{-2k} \binom{3n-k+2}{2} \\ &= 3^{-6}\|D^{\underline{2}}u\|_2 2^{-2(2n+|\underline{s}|)} \left(18n^2+42n+28-2^{-2n}\right).\end{aligned}$$

Similarly for the former sum one obtains

$$\sum_{k=n+1}^{2n-1} \sum_{|\underline{i}|=k \,\&\, \underline{i}\le\underline{n}} \|u^h_{\underline{i}+\underline{s}}\| \le 3^{-3}\|D^{\underline{2}}u\|_2 2^{-2|\underline{s}|} \sum_{k=n+1}^{2n-1} 2^{-2k} \sum_{|\underline{i}|=k \,\&\, \underline{i}\le\underline{n}} 1 .$$

It is straightforward to check that for $n < k < 2n$ one has

$$\sum_{|\underline{i}|=k \,\&\, \underline{i}\le\underline{n}} 1 = 3nk - k^2 + 1 - \frac{3}{2}n(n-1) .$$

Further,

$$\begin{aligned}&\sum_{k=n+1}^{2n-1} 2^{-2k} \left(3nk - k^2 + 1 - \frac{3}{2}n(n-1)\right) \\ &\qquad = 3^{-3}2^{-2n}\left(\frac{9}{2}(1-2^{2-2n})n^2 + \left(\frac{51}{2} - 42\cdot 2^{-2n}\right)n - 11 - 2^{4-2n}\right).\end{aligned}$$

Multiplying by $3^{-3}\|D^{\underline{2}}u\|_2 2^{-2|\underline{s}|}$ and adding to the other sum one obtains

$$\sum_{|\underline{i}|>n \,\&\, \underline{i}\le\underline{n}} \|u^h_{\underline{i}+\underline{s}}\| \le 3^{-6}\|D^{\underline{2}}u\|_2 2^{-2(n+|\underline{s}|)} \left(\frac{1}{2}(9n^2+51n-22) + 12\cdot 2^{-2n} - 2^{-4n}\right),$$

which is the desired result. □

As before we can extend these results to the error $\|u - u^t_{n,\underline{s}}\|_2$ via the triangle inequality. This result may also be extended to the L_∞ and L_1 norms by using the corresponding bounds of the $u^h_{\underline{i}}$ from [22]. Notice that n increases, the difference between the full grid solution $u_{\underline{f}} = u_{\underline{s}+\underline{n}}$ and $u^t_{n,\underline{s}}$ decreases at a rate $\mathcal{O}(2^{-2n}n^{d-1})$, i.e. the same rate that classical combinations converge to the true solution. Additionally if n is fixed and $\underline{s}$ increases then convergence is second order with respect to $|\underline{s}|$.

We also consider the error of truncated combinations when the $u_{\underline{i}}$ satisfy the error splitting model (2.22). Here we extend Theorem 2.27 to truncated combinations.

Theorem 4.6. *For all $\underline{i} \in \mathbb{N}^d$ let $u_{\underline{i}} : [0,1]^d \mapsto \mathbb{R}$ be approximations to $u : [0,1]^d \mapsto \mathbb{R}$ satisfying the error splitting* (2.22)*, that is*

$$u - u_{\underline{i}} = \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} v_{j_1,\dots,j_m}(h_{i_{j_1}},\dots,h_{i_{j_m}}) h_{i_{j_1}}^p \cdots h_{i_{j_m}}^p ,$$

with all of the $|v_{j_1,\dots,j_m}|$ bounded by a positive constant K. Let $u_{n,\underline{s}}^t$ be the truncated combination defined by (4.1)*, then*

$$|u - u_{n,\underline{s}}^t| \leq K 2^{-pn} \binom{n+d-1}{d-1} (1+2^p)^{d-1} \left(-1 + \prod_{k=1}^{d} (1 + 2^{-ps_k}) \right) .$$

Proof. The proof follows the same procedure as the proof of Theorem 2.27. The only difference is that the $h_{i_{j_1}} \cdots h_{i_{j_m}}$ terms become

$$h_{s_{j_1}+i_{j_1}} \cdots h_{s_{j_m}+i_{j_m}} = 2^{-s_{j_1}-\dots-s_{j_m}} h_{i_{j_1}} \cdots h_{i_{j_m}} .$$

By carrying this factor through the steps of the previous proof then we notice that at the line prior to (2.26) one has

$$|u - u_n^c| \leq 2^{-pn} \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} K \binom{n+m-1}{m-1} (1+2^p)^{m-1} ,$$

which for the truncated combination technique becomes

$$|u - u_{n,\underline{s}}^t| \leq 2^{-pn} \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} K 2^{-p(s_{j_1}+\dots+s_{j_m})} \binom{n+m-1}{m-1} (1+2^p)^{m-1} .$$

Bounding each $\binom{n+m-1}{m-1}(1+2^p)^{m-1}$ by the $m = d$ term $\binom{n+d-1}{d-1}(1+2^p)^{d-1}$ one obtains

$$\begin{aligned} |u - u_{n,\underline{s}}^t| &\leq K 2^{-pn} \binom{n+d-1}{d-1} (1+2^p)^{d-1} \sum_{m=1}^{d} \sum_{\substack{\{j_1,\dots,j_m\} \\ \subset\{1,\dots,d\}}} 2^{-p(s_{j_1}+\dots+s_{j_m})} \\ &= K 2^{-pn} \binom{n+d-1}{d-1} (1+2^p)^{d-1} \left(-1 + \prod_{k=1}^{d} (1 + 2^{-ps_k}) \right) . \end{aligned}$$

which is the desired result. ☐

We notice that when the $\underline{s}$ is isotropic (i.e. $s_1 = s_2 = \cdots = s_d$) then the bound reduces to

$$|u - u^t_{n,\underline{s}}| \leq K 2^{-pn} \binom{n+d-1}{d-1} (1 + 2^p)^{d-1} \left((1 + 2^{-ps_1})^d - 1\right) .$$

When $\underline{s} = \underline{0}$ then $u^t_{n,\underline{0}} = u^c_n$ and we obtain the bound $|u - u^c_n| \leq K 2^{-pn} \binom{n+d-1}{d-1} (1 + 2^p)^{d-1}(2^d - 1)$ which is weaker than that of Theorem 2.27 for $n > d - 1$.

4.2 Dimension Adaptive Sparse Grids

In this section we review the paper *Adaptive Sparse Grids* [70] in detail. The use of lattices of projections in the study of the combination technique was a nice idea and a powerful tool for extending the original notion of the combination technique. We extend upon the proofs in the paper making some clarifications as we go. Additionally we extend upon the existing work and obtain some new results. In particular, the original paper presented a procedure for calculating the updated coefficients. We quantify this result showing explicitly what the coefficient updates are. This allows for much faster computation of the coefficients and reveals more about the nature of adaptive sparse grids, namely the relation to the inclusion-exclusion principle. Further, we derive error formulae for adaptive sparse grids for both the classical interpolation of $u \in H^2_{0,\mathrm{mix}}$ and for $u_{\underline{i}}$ satisfying the error splitting model. Much of this section appears in the papers [64, 65].

In this thesis we consider lattice to be a partially ordered set $(\mathcal{L}, \leq)$ for which every two elements $a, b \in \mathcal{L}$ have a unique greatest lower bound, $a \wedge b$, and a unique least upper bound, $a \vee b$. It is well-known that a partially ordered set which is a lattice is equivalent to the algebraic lattice $(\mathcal{L}; \wedge, \vee)$ in which the binary operations $\wedge$ and $\vee$ acting on the non-empty set $\mathcal{L}$ are idempotent, commutative, associative and satisfy $a \wedge (a \vee b) = a \vee (a \wedge b) = a$ for all $a, b \in \mathcal{L}$. The natural definition of $\wedge$ and $\vee$ given a partially ordered set $(\mathcal{L}, \leq)$ is $a \wedge b = \inf\{a, b\}$ and $a \vee b = \sup\{a, b\}$.

We consider vector spaces which are tensor products $V = V^1 \times \cdots \times V^d$ and each component V^k ($k \in \{1, \dots, d\}$) is hierarchical, that is,

$$V^k_0 \subset V^k_1 \subset \cdots \subset V^k_{m_k} = V^k \,.$$

Additionally we assume that each $V^k_0 \neq \{0\}$ and each $V^k = V^k_{m_k}$ is finite dimensional. For the tensor product formulation we write $V_{\underline{i}} = V^1_{i_1} \times \cdots \times V^d_{i_d}$ where $\underline{i}$ is a multi-index $(i_1, \dots, i_d) \in \mathbb{N}^d$. We claim that the resulting function spaces form a lattice as we will demonstrate. An example of such spaces are the spaces of piecewise linear functions $V_{\underline{i}}$ defined in Section 2.1.

We define an ordering on these spaces given by $V^k_i \leq V^k_j$ iff $V^k_i \subseteq V^k_j$ for $i, j \in \mathbb{N}$. We see that this partial ordering relates to a natural ordering of the indices $V^k_i \subseteq V^k_j \Leftrightarrow i \leq j$. The greatest lower bound of two such spaces is then given by $V^k_i \wedge V^k_j = V^k_i \cap V^k_j = V^k_{\min\{i,j\}}$. Note that the natural ordering of $\{0, 1, \dots, m_k\} \subset \mathbb{N}$ leads to the greatest lower bound $i \wedge j = \min\{i, j\}$ such that we can write $V^k_i \wedge V^k_j = V^k_{i \wedge j}$. Similarly we have the least upper bound given by

$V_i^k \vee V_j^k$. In general this is given by the linear span of the V_i^k and V_j^k. In this case the hierarchical/nested structure of the V^k means that $V_i^k \vee V_j^k = V_i^k \cup V_j^k = V_{\max\{i,j\}}^k = V_{i\vee j}^k$. Thus we see that the lattice structure on the V_i^k can be viewed as being lifted from $\mathbb{N}$ via $V_i^k \leq V_j^k \Leftrightarrow i \leq j$, $V_i^k \wedge V_j^k = V_{i\wedge j}^k$ and $V_i^k \vee V_j^k = V_{i\vee j}^k$.

Now we consider the partial ordering induced on the tensor product space V. Given $V_{\underline{i}}$ and $V_{\underline{j}}$ we say $V_{\underline{i}} \leq V_{\underline{j}}$ iff $V_{i_k} \leq V_{j_k}$ for $k = 1, \dots, d$, or equivalently $V_{\underline{i}} \subseteq V_{\underline{j}}$. The greatest lower bound can also be defined via the tensor product $V_{\underline{i}} \wedge V_{\underline{j}} := (V_{i_1}^1 \wedge V_{j_1}^1) \otimes \cdots \otimes (V_{i_d}^d \wedge V_{j_d}^d)$. Alternatively we can write this as $V_{\underline{i}} \wedge V_{\underline{j}} = V_{\min\{\underline{i},\underline{j}\}}$ where the min is taken component wise. Similarly we define the least upper bound as $V_{\underline{i}} \vee V_{\underline{j}} := (V_{i_1}^1 \vee V_{j_1}^1) \otimes \cdots \otimes (V_{i_d}^d \vee V_{j_d}^d)$ which can be simplified to $V_{\max\{\underline{i},\underline{j}\}}$ (with the max also taken component wise). Note that whilst $V_{\underline{i}} \wedge V_{\underline{j}} = V_{\underline{i}} \cap V_{\underline{j}}$ it is generally not true that $V_{\underline{i}} \vee V_{\underline{j}}$ is equal to $V_{\underline{i}} \cup V_{\underline{j}}$. Just as the lattice structure on the V_i^k can be viewed as having been lifted from the natural lattice on $\mathbb{N}$, the same is also true of the $V_{\underline{i}}$. Consider the partial ordering on $\underline{i}, \underline{j} \in \mathbb{N}^d$ defined by $\underline{i} \leq \underline{j}$ iff $i_k \leq j_k$ for $k = 1, \dots, d$. The addition of the binary relations $\underline{i} \wedge \underline{j} := \min\{\underline{i}, \underline{j}\}$ and $\underline{i} \vee \underline{j} := \max\{\underline{i}, \underline{j}\}$ describes a lattice on $\mathbb{N}^d$. Thus we observe the 'lifting' of the lattice structure via $V_{\underline{i}} \leq V_{\underline{j}} \Leftrightarrow \underline{i} \leq \underline{j}$, $V_{\underline{i}} \wedge V_{\underline{j}} = V_{\underline{i}\wedge\underline{j}}$ and $V_{\underline{i}} \vee V_{\underline{j}} = V_{\underline{i}\vee\underline{j}}$.

With these binary operations our nested function spaces now form a lattice. Similarly we have a corresponding lattice on the multi-indices in $\mathbb{N}^d$. Figure 4.2 depicts some elements of $\mathbb{N}^d$ and describes their relations in the lattice. From here we will stick to discussing the lattice as applied to $\mathbb{N}^d$ acknowledging that this can be lifted to the corresponding function space lattice. We continue with a few more definitions.

An element $\underline{j} \in \mathbb{N}^d$ is said to cover another element $\underline{i} \in \mathbb{N}^d$ if $\underline{i} < \underline{j}$ and there is no $\underline{l}$ such that $\underline{i} < \underline{l} < \underline{j}$ (here $<$ means $\leq$ and $\neq$), see for example [36]. Equivalently, $\underline{j}$ covers $\underline{i}$ if $j_k = i_k + 1$ for exactly one $k \in \{1, \dots, d\}$ and $j_r = i_r$ for $r \neq k$. We use the notation $\underline{i} \prec \underline{j}$ to denote that $\underline{j}$ covers $\underline{i}$. Similarly, $V_{\underline{j}}$ covers $V_{\underline{i}}$ if $V_{\underline{i}} \leq V_{\underline{j}}$ and there is no $V_{\underline{l}}$ such that $V_{\underline{i}} < V_{\underline{l}} < V_{\underline{j}}$. The lattice of multi-indices in $\mathbb{N}^d$ is also graded when equipped with the rank function $r(\underline{i}) = |\underline{i}|$. Analogously the lattice $V_{\underline{i}}$ is graded when equipped with the rank function $r(V_{\underline{i}}) = |\underline{i}|$.

We will consider a family of projections $P_{i_k}^k : V^k \to V_{i_k}^k$. Taking the tensor product provides the projection $P_{\underline{i}} = \bigotimes_k P_{i_k}^k : V \to V_{\underline{i}}$. The existence of such projections is given by the following proposition.

Proposition 4.7 ([70]). *For every lattice space generated from a tensor product of hierarchical spaces we have:*

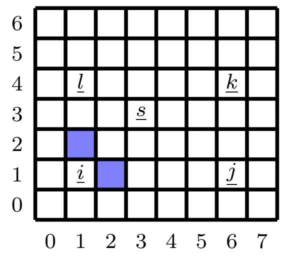


Figure 4.2: *Here we depict several elements of $\mathbb{N}^2$ and describe their relations with respect to the natural lattice described in the text. Oberve that $\underline{k} \geq \underline{j} \geq \underline{i}$, $\underline{k} \geq \underline{l} \geq \underline{i}$ and $\underline{k} \geq \underline{s} \geq \underline{i}$. Note that $\underline{s}$ is neither $\leq$ nor $\geq$ either of $\underline{j}$ and $\underline{l}$. Additionally one has $\underline{j} \wedge \underline{l} = \underline{i}$, $\underline{j} \vee \underline{l} = \underline{k}$, $\underline{s} \wedge \underline{i} = \underline{i}$ and $\underline{s} \vee \underline{k} = \underline{k}$. The two elements highlighted in blue are the two possible covering elements of $\underline{i}$. The diagram and relations described can also be interpreted as applying to the lattice on $V_{\underline{i}}$.*

- *there are linear operators $P_{\underline{i}}$ on V with range $R(P_{\underline{i}}) = V_{\underline{i}}$ and $P_{\underline{i}}P_{\underline{j}} = P_{\underline{i}\wedge\underline{j}}$.*
- *Consequently $P_{\underline{i}}P_{\underline{i}} = P_{\underline{i}}$ and $P_{\underline{i}}P_{\underline{j}} = P_{\underline{j}}P_{\underline{i}}$.*

Proof. The results are an immediate consequence of the tensor product construction of the lattice. □

Let I be a subset of the lattice of multi-indices on $\mathbb{N}^d$. We say I is a downset if

$$\underline{i} \in I \text{ and } \underline{j} \leq \underline{i} \Rightarrow \underline{j} \in I\,.$$

Given $J \subset \mathbb{N}^d$ we use the notation $J\!\downarrow$ to denote the smallest downset that contains J. Consider $\mathcal{P}(\mathbb{N}^d)$, i.e. the power set of the set of all multi-indices, and let $\mathcal{D}(\mathbb{N}^d)$ be the restriction of the power set to contain only finite downsets. We can define a partial ordering and binary relations on the set of downsets.

Definition 4.8. Given $I, J \in \mathcal{D}(\mathbb{N}^d)$ then we define the partial ordering

$$I \leq J \Leftrightarrow I \subseteq J$$

Additionally we define the binary relations

$$I \wedge J := I \cap J$$
$$I \vee J := I \cup J\,.$$

This leads us to the following lemma.

Lemma 4.9. *$\mathcal{D}(\mathbb{N}^d)$ with the partial ordering and binary operations defined in Definition 4.8 is a lattice.*

Proof. We need only show that given any $I, J \in \mathcal{D}(\mathbb{N}^d)$ then $I \wedge J \in \mathcal{D}(\mathbb{N}^d)$ and $I \vee J \in \mathcal{D}(\mathbb{N}^d)$.

Let $\underline{i} \in I \wedge J = I \cap J$, then $\underline{i} \in I$ and $\underline{i} \in J$. It follows for each $\underline{j} \in \mathbb{N}^d$ such that $\underline{j} \leq \underline{i}$ then $\underline{j} \in I$ and $\underline{j} \in J$ since I and J are downsets and therefore $\underline{j} \in I \wedge J$. As a consequence $I \wedge J$ is a downset.

Similarly it can be shown that $I \vee J$ is also a downset. □

We also have the cover relation $I \prec J$ iff $J = I \cup \{\underline{i}\}$ for some $\underline{i} \notin I$ for which $\underline{j} \prec \underline{i} \Rightarrow \underline{j} \in I$ for all $\underline{j} \in \mathbb{N}^d$ (or equivalently $\underline{j} < \underline{i} \Rightarrow \underline{j} \in I$ for all $\underline{j} \in \mathbb{N}^d$). Figure 4.3 depicts several elements of $\mathbb{N}^2$ and describes their relations.

Just as the lattice on $\mathbb{N}^d$ can be lifted to a lattice on $\{V_{\underline{i}}\}_{\underline{i} \in \mathbb{N}^d}$ we can lift the lattice on $\mathcal{D}(\mathbb{N}^d)$ to the so called combination space lattice

$$V_I := \sum_{\underline{i} \in I} V_{\underline{i}} \,,$$

for $I \in \mathcal{D}(\mathbb{N}^d)$. It is straightforward to show that V_I is a downset if I is itself a downset, and furthermore $V_J{\downarrow} = V_{J\downarrow}$. Furthermore we can define the partial ordering $V_I \leq V_J$ iff $I \leq J$ and the binary relations $V_I \wedge V_J = V_{I \wedge J}$ and $V_I \vee V_J = V_{I \vee J}$. The result is therefore a lattice on $\{V_I\}_{I \in \mathcal{D}(\mathbb{N}^d)}$. We also have the covering relation $V_I \prec V_J$ iff $I \prec J$. This brings us to a second proposition.

Proposition 4.10 ([70]). *Let the lattices $V_{\underline{i}}$ have the projections $P_{\underline{i}}$ as in Proposition 4.7, then for $I, J \in \mathcal{D}(\mathbb{N}^d)$ there are linear operators P_I on V with range $R(P_I) = V_I$ such that $P_I P_J = P_{I \cap J}$. Conversely, if P_I is a family of projections with these properties, then $P_{\underline{i}} := P_{\{\underline{i}\}\downarrow}$ defines a family of projections as in Proposition 4.7.*

This is as in [70] and is closely linked to the lattice I described on $\mathcal{D}(\mathbb{N}^d)$. An extended proof is given here.

Proof. We define the linear operators

$$P_I = 1 - \prod_{\underline{i} \in I} (1 - P_{\underline{i}}) \,.$$

Now using the fact that $P_{\underline{i}} P_{\underline{j}} = P_{\underline{j}}$ if $\underline{j} \leq \underline{i}$ we claim that $P_I = 1 - \prod_{\underline{i} \in \max I} (1 - P_{\underline{i}})$, where $\max I$ are the maximal elements of I, i.e. $\underline{i} \in \max I$ if there is no $\underline{l} \in I \backslash \{\underline{i}\}$ such that $\underline{i} \leq \underline{l}$. Let $\underline{l} \in I \backslash \max I$, then

$$\begin{aligned} 1 - \prod_{\underline{i} \in I} (1 - P_{\underline{i}}) &= 1 - (1 - P_{\underline{l}}) \prod_{\underline{i} \in I \backslash \{\underline{l}\}} (1 - P_{\underline{i}}) \\ &= 1 - \prod_{\underline{i} \in I \backslash \{\underline{l}\}} (1 - P_{\underline{i}}) + P_{\underline{l}} \prod_{\underline{i} \in I \backslash \{\underline{l}\}} (1 - P_{\underline{i}}) \,. \end{aligned}$$

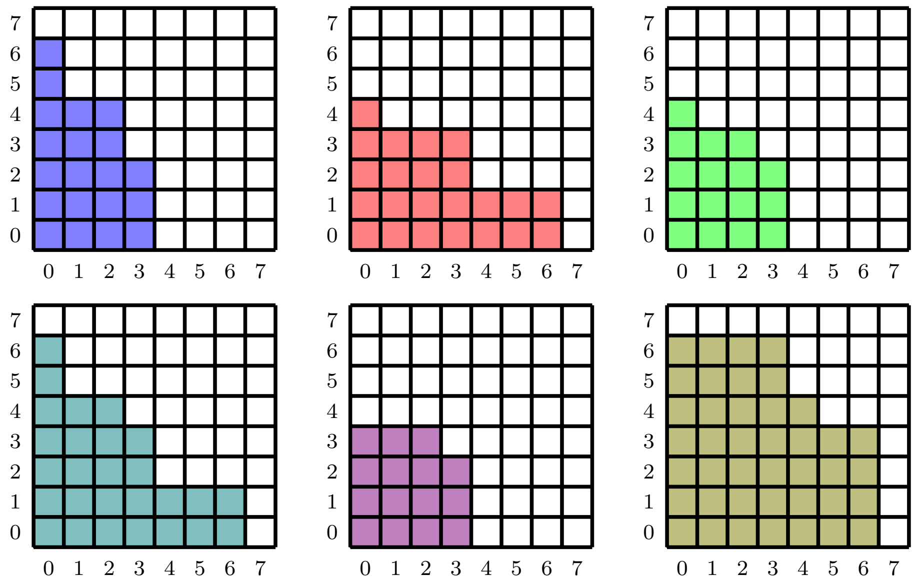


Figure 4.3: *Here we depict a few elements of $\mathcal{D}(\mathbb{N}^2)$ and describe their relations with respect to the lattice described in the text. Clockwise from the top left let us denote the sets by A, B, C, D, E and F respectively. One has $E \leq C \leq A \leq F \leq D$, $C \leq B \leq F$, $A \wedge B = C$ and $A \vee B = F$. Additionally, C is a cover of E. The diagrams and relations described can also be interpreted as applying to the lattice on $\{V_I\}_{I \in \mathcal{D}(\mathbb{N}^2)}$.*

Now examining the last product we have

$$\begin{aligned} P_{\underline{l}} \prod_{\underline{i} \in I \setminus \{\underline{l}\}} (1 - P_{\underline{i}}) &= \prod_{\underline{i} \in I \setminus \{\underline{l}\}} P_{\underline{l}}(1 - P_{\underline{i}}) \\ &= \prod_{\underline{i} \in I \setminus \{\underline{l}\}} (P_{\underline{l}} - P_{\underline{l} \wedge \underline{i}}) \end{aligned}$$

but since $\underline{l} \notin \max I$ then there exists $\underline{i}' \in \max I$ such that $\underline{l} \leq \underline{i}'$ and therefore $P_{\underline{l} \wedge \underline{i}'} = P_{\underline{l}}$. Hence $\prod_{\underline{i} \in I \setminus \{\underline{l}\}} (P_{\underline{l}} - P_{\underline{l} \wedge \underline{i}}) = 0$ and

$$1 - \prod_{\underline{i} \in I} (1 - P_{\underline{i}}) = 1 - \prod_{\underline{i} \in I \setminus \{\underline{l}\}} (1 - P_{\underline{i}}) \,.$$

By repeating this argument on all elements in $I \backslash \max I$ we obtain $P_I = 1 - \prod_{\underline{i} \in \max I} (1 - P_{\underline{i}})$.

Now, by expanding the product over the maximal elements and using the equality $P_{\underline{i}} P_{\underline{j}} = P_{\underline{i} \wedge \underline{j}}$ it follows that

$$P_I = \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}}$$

for some $c_{\underline{i}}$, referred to as combination coefficients, which are zero if $\underline{i}$ is not in the sub-lattice generated by $\max I$. Note that if $\underline{i} \in \max I$ then $c_{\underline{i}} = 1$. Thus the range of P_I is V_I. Let $Q = P_I P_J - P_{I\cap J}$. The range is $V_{I\cap J}$ which is in the null space of Q. Now P_I is a projection mapping elements of V_I onto themselves and so $Q^2 = Q$ and thus Q=0. The converse follows directly. $\square$

Corollary 3 of [70] tells one how to update the coefficients when a new element if added to the downset. Unfortunately the result does not make it particularly clear how to compute the updated coefficients. We clarify this in the alternate version of the corollary below.

Corollary 4.11 ([70])**.** *Let $J = I \cup \{\underline{j}\}$ be a covering element of I and let P_I be the family of projections as in the previous proposition and $P_{\underline{i}} = P_{\downarrow\underline{i}}$. Then one has:*

$$P_J - P_I = \sum_{\underline{i}\in J} d_{\underline{i}} P_{\underline{i}}$$

where $d_{\underline{j}} = 1$ and for $\underline{i} \in I$ we have

$$d_{\underline{i}} = -\sum_{\underline{l}\in I_{\underline{i}|\underline{j}}} c_{\underline{l}}$$

with $I_{\underline{i}|\underline{j}} := \{\underline{l} \in I : \underline{j} \wedge \underline{l} = \underline{i}\}$.

Proof. Notice that

$$\begin{aligned}
P_J - P_I &= \left(1 - (1-P_{\underline{j}})\prod_{\underline{i}\in I}(1-P_{\underline{i}})\right) - \left(1 - \prod_{\underline{i}\in I}(1-P_{\underline{i}})\right)\\
&= P_{\underline{j}}\prod_{\underline{i}\in I}(1-P_{\underline{i}})\\
&= P_{\underline{j}} - P_{\underline{j}}\left(1 - \prod_{\underline{i}\in I}(1-P_{\underline{i}})\right) = P_{\underline{j}} - P_{\underline{j}}P_I\,.
\end{aligned}$$

Therefore with $P_I = \sum_{\underline{i}\in I} c_{\underline{i}} P_{\underline{i}}$ one has

$$\begin{aligned}
P_J - P_I &= P_{\underline{j}} - P_{\underline{j}}\sum_{\underline{i}\in I} c_{\underline{i}} P_{\underline{i}}\\
&= P_{\underline{j}} - \sum_{\underline{i}\in I} c_{\underline{i}} P_{\underline{j}\wedge\underline{i}}\\
&= P_{\underline{j}} - \sum_{\underline{i}\in I} P_{\underline{i}} \sum_{\{\underline{l}\in I:\underline{l}\wedge\underline{j}=\underline{i}\}} c_{\underline{l}} = \sum_{\underline{i}\in J} d_{\underline{i}} P_{\underline{i}}
\end{aligned}$$

as required. $\square$

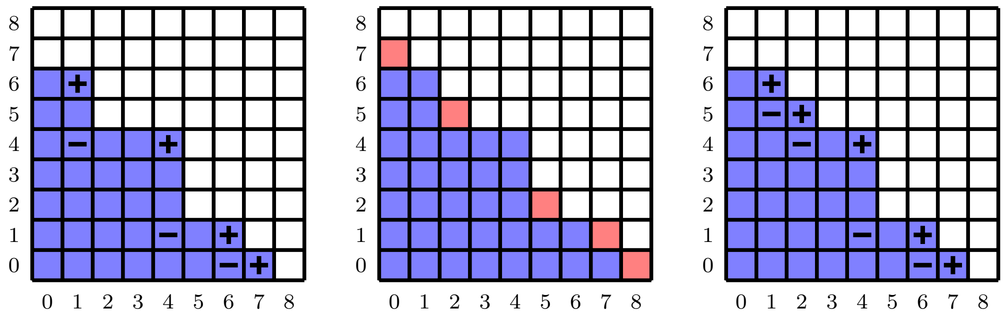


Figure 4.4: *Here we depict an iteration of adaptive sparse grid algorithm in $d = 2$ dimensions. The x and y axes denote the i_1 and i_2 components of $\underline{i} \in \mathbb{N}^2$. On the left is a downset I (shaded blue) and the corresponding combination coefficients ($+$ for $+1$ and $-$ for -1). In the middle we have identified the covering elements of I in red. On the right a covering element $\underline{j}$ has been chosen and the corresponding downset $J = I \cup \{\underline{j}\}$ is shown (shaded blue) with the new combination coefficients.*

As a result, if we have a solution to a problem using the combination P_I and we add a solution from another grid $V_{\underline{j}}$ such that the new lattice is $J = I \cup \{\underline{j}\}$ which is a covering element of I then the combination coefficients are given by:

$$P_J = P_{\underline{j}} + \sum_{\underline{i} \in I} (c_{\underline{i}} + d_{\underline{i}}) P_{\underline{i}}$$

where the $d_{\underline{i}}$'s are given by the corollary. An example of this process is depicted in Figure 4.4. This completes our review of [70] and we will now extend upon this work, but first we make a couple of brief remarks.

Remark 4.12. An important observation to be made is that combination coefficients are uniquely determined by a set of maximal elements. In particular, given a downset I then the coefficients are obtained by expanding

$$P_I = 1 - \prod_{\underline{i} \in \max I} (1 - P_{\underline{i}}) \to \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}} .$$

In particular there is a one-to-one correspondence between sets of maximal elements and sets of combination coefficients.

Remark 4.13. We also observe that the assumption that I, J are downsets can be replaced with the assumption that I, J are sets which are closed under $\wedge$. Such sets are known as lower semi-lattices. This is a consequence of the coefficients being non-zero only for $\underline{i}$ in the closure of the maximal elements under $\wedge$ which we denote by $(\max I)_\wedge$. In particular this means that we can reduce the complexity of computing a combination by only computing projections $P_{\underline{i}}$ for $\underline{i} \in (\max I)_\wedge \subset I$ rather than the whole downset $I{\downarrow}$. Note that the assumption that $J = I \cup \{\underline{j}\}$ for $I, J \in \mathcal{D}(\mathbb{N}^d)$ in Corollary 4.11 becomes $J{\downarrow} = I{\downarrow} \cup \{\underline{j}\}$ for $I, J \subset \mathbb{N}^d$ closed under $\wedge$.

We now introduce projections onto hierarchical surpluses which will be an important tool in the remainder of this section.

Definition 4.14. For $k = 1, \dots, d$ let $i_k \in \mathbb{N}$ and $Q^k_{i_k} : V^k \to V^k$ be defined as $Q^k_{i_k} := P^k_{i_k} - P^k_{i_k-1}$ (where $P^k_{i_k-1} := 0$ if $i_k - 1 < 0$).

We now give some basic properties of the $Q^k_{i_k}$.

Lemma 4.15. *Let $Q^k_{i_k}$ be as defined in Definition 4.14, then*

1. *$Q^k_{i_k}$ has co-domain $V^k_{i_k}$,*
2. *$P^k_{j_k} Q^k_{i_k} = Q^k_{i_k} P^k_{j_k} = 0$ if $j_k < i_k$,*
3. *$P^k_{j_k} Q^k_{i_k} = Q^k_{i_k} P^k_{j_k} = Q^k_{i_k}$ if $j_k \geq i_k$,*
4. *$Q^k_{i_k} Q^k_{i_k} = Q^k_{i_k}$,*
5. *$Q^k_{j_k} Q^k_{i_k} = 0$ if $i_k \neq j_k$*
6. *$P^k_{i_k} = \sum_{j_k=0}^{i_k} Q^k_{j_k}$*

Proof. The first is immediate as $P^k_{i_k}$ and $P^k_{i_k-1}$ have range $V^k_{i_k}$ and $V^k_{i_k-1} \subset V^k_{i_k}$ respectively. For the second and third properties we observe

$$P^k_{j_k} Q^k_{i_k} = P^k_{j_k} P^k_{i_k} - P^k_{j_k} P^k_{i_k-1} = \begin{cases} P^k_{j_k} - P^k_{j_k} = 0 & \text{for } j_k < i_k \,, \\ P^k_{i_k} - P^k_{i_k-1} = Q^k_{i_k} & \text{for } j_k \geq i_k \,. \end{cases}$$

It follows that

$$Q^k_{j_k} Q^k_{i_k} = (P^k_{j_k} - P^k_{j_k-1}) Q^k_{i_k} = \begin{cases} Q^k_{i_k} - 0 = Q^k_{i_k} & \text{for } j_k = i_k \,, \\ Q^k_{i_k} - Q^k_{i_k} = 0 & \text{for } j_k > i_k \,, \\ 0 - 0 = 0 & \text{for } j_k < i_k \,, \end{cases}$$

which corresponds to the fourth and fifth properties. The final property is a result of the telescoping sum

$$\sum_{j_k=0}^{i_k} Q^k_{j_k} = \sum_{j_k=0}^{i_k} P^k_{j_k} - P^k_{j_k-1} = P^k_{i_k} - P^k_{-1} = P^k_{i_k} \,.$$

□

As with the $P_{\underline{i}}$ we define $Q_{\underline{i}} := \bigotimes_k Q^k_{i_k}$. This leads us to the following lemma

Lemma 4.16. *Let* $Q^k_{i_k} := P^k_{i_k} - P^k_{i_k-1}$ *and* $Q_{\underline{i}} := \bigotimes_k Q^k_{i_k}$*, then*

$$Q_{\underline{i}} = \sum_{\underline{j} \leq \underline{1}} (-1)^{|\underline{j}|} P_{\underline{i}-\underline{j}}$$

with $P_{\underline{i}-\underline{j}} = 0$ *if* $i_k - j_k < 0$ *for some* $k \in \{1, \dots, d\}$.

Proof. One has

$$\begin{aligned} Q_{\underline{i}} = \bigotimes_{k=1}^{d} Q^k_{i_k} &= \bigotimes_{k=1}^{d} (P^k_{i_k} - P^k_{i_k-1}) \\ &= \sum_{\underline{j} \leq \underline{1}} (-1)^{|\underline{j}|} \bigotimes_{k=1}^{d} P^k_{i_k - j_k} = \sum_{\underline{j} \leq \underline{1}} (-1)^{|\underline{j}|} P_{\underline{i}-\underline{j}} \,, \end{aligned}$$

as required. Since $P^k_{i_k-1} := 0$ if $i_k - 1 < 0$ one obtains $P_{\underline{i}-\underline{j}} = 0$ if $i_k - j_k < 0$ for some $k \in \{1, \dots, d\}$. □

Additionally we have that the $P_{\underline{i}}$ is equal to a sum of the $Q_{\underline{j}}$ for $\underline{j} \leq \underline{i}$.

Lemma 4.17. *With* $P_{\underline{i}}$ *and* $Q_{\underline{i}}$ *as previously defined one has*

$$P_{\underline{i}} = \sum_{(\underline{0} \leq)\, \underline{j} \leq \underline{i}} Q_{\underline{j}} \,.$$

Proof. We note that

$$\begin{aligned} \sum_{(\underline{0} \leq)\, \underline{j} \leq \underline{i}} Q_{\underline{j}} &= \sum_{j_1=0}^{i_1} \cdots \sum_{j_d=0}^{i_d} \bigotimes_{k=1}^{d} (P^k_{j_k} - P^k_{j_k-1}) \\ &= \bigotimes_{k=1}^{d} \sum_{j_k=0}^{i_k} (P^k_{j_k} - P^k_{j_k-1}) \\ &= \bigotimes_{k=1}^{d} P^k_{i_k} = P_{\underline{i}} \end{aligned}$$

as required. □

A few more properties of $Q_{\underline{i}}$ will also be useful.

Lemma 4.18. *With* $P_{\underline{i}}$ *and* $Q_{\underline{i}}$ *as previously defined one has*

1. $Q_{\underline{i}} Q_{\underline{i}} = Q_{\underline{i}}$,

2. $Q_{\underline{i}} Q_{\underline{j}} = Q_{\underline{j}} Q_{\underline{i}} = 0$ *for* $\underline{j} \neq \underline{i}$,

3. *$P_{\underline{i}}Q_{\underline{j}} = Q_{\underline{j}}P_{\underline{i}} = Q_{\underline{j}}$ for $\underline{j} \leq \underline{i}$,*

4. *$P_{\underline{i}}Q_{\underline{j}} = Q_{\underline{j}}P_{\underline{i}} = 0$ for $\underline{j} \not\leq \underline{i}$ (that is $\underline{j} \in \mathbb{N}^d \backslash \{\underline{i}\}\!\downarrow$).*

Proof. The first two are a direct consequence of the analogous results shown for the $Q^k_{i_k}$'s in Lemma 4.15 and the latter two are a consequence of the first two combined with Lemma 4.17. □

Given a downset $I \in \mathcal{D}(\mathbb{N}^d)$ let us define $Q_I := \sum_{\underline{i}\in I} Q_{\underline{i}}$. Given that $P_{\underline{i}} = \sum_{\underline{j}\leq\underline{i}} Q_{\underline{j}}$ it would be reasonable to expect that $Q_I = P_I$ and indeed this is the subject of the following proposition.

Proposition 4.19. *Let $I \in \mathcal{D}(\mathbb{N}^d)$, then $P_I = Q_I$.*

Proof. We have that

$$\begin{aligned} P_I &= 1 - \prod_{\underline{i}\in\max I}(1 - P_{\underline{i}}) \\ &= 1 - \sum_{J\subset\max I}(-1)^{|J|}\prod_{\underline{j}\in J} P_{\underline{j}} \\ &= \sum_{\substack{J\subset\max I \\ J\neq\emptyset}}(-1)^{|J|+1}\sum_{\underline{j}\leq\wedge J} Q_{\underline{j}}\,, \end{aligned}$$

where $|J|$ is the number of elements of J and $\wedge J$ is the greatest lower bound over all elements of J (that is if $J = \{\underline{j}_1, \dots, \underline{j}_{|J|}\}$ then $\wedge J = \underline{j}_1 \wedge \cdots \wedge \underline{j}_{|J|}$). This last line reduces to

$$P_I = \sum_{\underline{j}\in I} d_{\underline{i}} Q_{\underline{j}}$$

for some coefficients $d_{\underline{i}}$. We are required to show that $d_{\underline{i}} = 1$ for all $\underline{i} \in I$. Consider $Q_{\underline{i}}P_I$, using Lemma 4.18 we have

$$Q_{\underline{i}}P_I = \sum_{\underline{j}\in I} d_{\underline{j}} Q_{\underline{i}} Q_{\underline{j}} = d_{\underline{i}} Q_{\underline{i}}\,.$$

Additionally we note that

$$Q_{\underline{i}} \sum_{\underline{j}\leq\wedge J} Q_{\underline{j}} = \begin{cases} Q_{\underline{i}} & \text{if } \underline{i} \leq \wedge J \\ 0 & \text{otherwise}\,. \end{cases}$$

Further, $\underline{i} \leq \wedge J$ if and only if $\underline{i} \leq \underline{j}$ for all $\underline{j} \in J$. Let $J_{\underline{i}} \subset \max I$ be the largest set such that $\underline{i} \leq \underline{j}$ for all $\underline{j} \in J_{\underline{i}}$. Thus

$$\begin{aligned}
Q_{\underline{i}} P_I = \sum_{\substack{J \subset \max I \\ J \neq \emptyset}} (-1)^{|J|+1} \sum_{\underline{j} \leq \wedge J} Q_{\underline{i}} Q_{\underline{j}} &= \sum_{m=1}^{|J_{\underline{i}}|} \sum_{\{\underline{j}_1, \dots, \underline{j}_m\} \subset J_{\underline{i}}} (-1)^{m+1} Q_{\underline{i}} \\
&= Q_{\underline{i}} \sum_{m=1}^{|J_{\underline{i}}|} \binom{|J_{\underline{i}}|}{m} (-1)^{m+1} \\
&= Q_{\underline{i}} \left(1 - \sum_{m=0}^{|J_{\underline{i}}|} \binom{|J_{\underline{i}}|}{m} (-1)^m \right) \\
&= Q_{\underline{i}} \left(1 - (1-1)^{|J_{\underline{i}}|} \right) = Q_{\underline{i}} \, .
\end{aligned}$$

Therefore $d_{\underline{i}} Q_{\underline{i}} = Q_{\underline{i}} P_I = Q_{\underline{i}}$ and hence $d_{\underline{i}} = 1$. Since the choice of $\underline{i} \in I$ was arbitrary the proof is complete. □

This brings us to a lemma which says something about the nature of the combination coefficients.

Lemma 4.20. *Given $I \in \mathcal{D}(\mathbb{N}^d)$ with corresponding projection*

$$P_I = \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}}$$

one has for each $\underline{l} \in I$

$$c_{\underline{l}} = 1 - \sum_{\underline{l} < \underline{j} \in I} c_{\underline{j}} \, .$$

Proof. From Proposition 4.19 one has that

$$\sum_{\underline{i} \in I} Q_{\underline{i}} = P_I = \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}} = \sum_{\underline{i} \in I} c_{\underline{i}} \sum_{\underline{j} \leq \underline{i}} Q_{\underline{j}} \, .$$

It follows that for $\underline{l} \in I$ one has

$$Q_{\underline{l}} = Q_{\underline{l}} P_I = \sum_{\underline{i} \in I} c_{\underline{i}} \sum_{\underline{j} \leq \underline{i}} Q_{\underline{l}} Q_{\underline{j}} = \sum_{\underline{l} \leq \underline{i} \in I} c_{\underline{i}} Q_{\underline{l}} \, .$$

Therefore

$$1 = \sum_{\underline{l} \leq \underline{j} \in I} c_{\underline{j}}$$

and re-arranging gives the desired result. □

Let χ_I be the characteristic function on multi-indices, that is

$$\chi_I(\underline{i}) := \begin{cases} 1 & \text{if } \underline{i} \in I, \\ 0 & \text{otherwise.} \end{cases}$$

We now give a useful result regarding the calculation of combination coefficients.

Proposition 4.21. *Let $I \in \mathcal{D}(\mathbb{N}^d)$ with corresponding projection*

$$P_I = \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}} \,,$$

then for each $\underline{i} \in I$ one has

$$c_{\underline{i}} = \sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} (-1)^{|\underline{j}-\underline{i}|} \chi_I(\underline{j}) \,.$$

Proof. From Lemma 4.20 we have that for $\underline{i} \in I$

$$1 = \sum_{\underline{i} \leq \underline{j} \in I} c_{\underline{j}} = \sum_{\underline{i} \leq \underline{j}} c_{\underline{j}} \chi_I(\underline{j}) \,.$$

It follows that for $\underline{i} \in \mathbb{N}^d$

$$\chi_I(\underline{i}) = \sum_{\underline{i} \leq \underline{j}} c_{\underline{j}} \chi_I(\underline{j}) \,.$$

We substitute this into the following

$$\begin{aligned} \sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} (-1)^{|\underline{j}-\underline{i}|} \chi_I(\underline{j}) &= \sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} (-1)^{|\underline{j}-\underline{i}|} \sum_{\underline{j} \leq \underline{l}} c_{\underline{l}} \chi_I(\underline{l}) \\ &= \sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} \left((-1)^{|j_1 - i_1|} \sum_{l_1 = j_1}^{\infty} \cdots (-1)^{|j_d - i_d|} \sum_{l_d = j_d}^{\infty} \right) c_{\underline{l}} \chi_I(\underline{l}) \\ &= \left(\sum_{l_1 = i_1}^{\infty} - \sum_{l_1 = i_1 + 1}^{\infty} \right) \cdots \left(\sum_{l_d = i_d}^{\infty} - \sum_{l_d = i_d + 1}^{\infty} \right) c_{\underline{l}} \chi_I(\underline{l}) \\ &= c_{\underline{i}} \chi_I(\underline{i}) \,, \end{aligned}$$

as required. □

We are now able to easily compute combination coefficients given an arbitrary downset I. Many of the coefficients are typically 0 as we will see from the following corollary.

Corollary 4.22. *Let $I \in \mathcal{D}(\mathbb{N}^d)$ with corresponding projection $P_I = \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}}$. Let $\underline{i} \in I$, if $\underline{i} + \underline{1} \in I$ then $c_{\underline{i}} = 0$.*

Proof. $\underline{i}+\underline{1}\in I$ implies that $\underline{j}\in I$ for all $\underline{j}\leq\underline{i}+\underline{1}$. Therefore using Proposition 4.21 we have

$$c_{\underline{i}} = c_{\underline{i}}\chi_I(\underline{i}) = \sum_{\underline{i}\leq\underline{j}\leq\underline{i}+\underline{1}} (-1)^{|\underline{i}-\underline{j}|}\chi_I(\underline{j}) = \sum_{\underline{i}\leq\underline{j}\leq\underline{i}+\underline{1}} (-1)^{|\underline{i}-\underline{j}|} = 0\,,$$

as required. □

In [70] an update formula was given when a covering element is added to a downset I. It turns out that the update coefficients have a very particular structure as the next lemma will demonstrate.

Lemma 4.23. *Let $I,J\in\mathcal{D}(\mathbb{N}^d)$ such that $I\prec J$. In particular, let $\underline{i}$ be the multi-index such that $J=I\cup\{\underline{i}\}$. Then*

$$P_J - P_I = \sum_{\underline{i}-\underline{1}\leq\underline{j}\leq\underline{i}} (-1)^{|\underline{i}-\underline{j}|}P_{\underline{j}}$$

where $P_{\underline{j}} := 0$ if any of the $j_k<0$.

Proof. Clearly we have $P_J-P_I=Q_{\underline{i}}$. Now we simply note that

$$\begin{aligned} Q_{\underline{i}} &= \sum_{\underline{j}\leq\underline{1}} (-1)^{|\underline{j}|}P_{\underline{i}-\underline{j}} \\ &= \sum_{\underline{i}-\underline{1}\leq\underline{j}\leq\underline{i}} (-1)^{|\underline{i}-\underline{j}|}P_{\underline{j}}\,, \end{aligned}$$

as required. □

This is quite a useful result. For example, in 2 dimensions when a covering element is added with $\underline{i}=(i_1,i_2)\geq(1,1)$ then only 4 coefficients need to be changed. Namely $c_{(i_1,i_2)}\mapsto 1$, $c_{(i_1-1,i_2)}\mapsto c_{(i_1-1,i_2)}-1$, $c_{(i_1,i_2-1)}\mapsto c_{(i_1,i_2-1)}-1$ and $c_{(i_1-1,i_2-1)}\mapsto c_{(i_1-1,i_2-1)}+1$. Similarly in d dimensions one only needs to change 2^d coefficients.

The following lemma shows that if the downset I is non-empty then the coefficients will sum to 1 which is essentially a consistency property for adaptive sparse grids.

Lemma 4.24. *If $I\in\mathcal{D}(\mathbb{N}^d)$ is non-empty, then the coefficients $c_{\underline{i}}$ corresponding to*

$$P_I = \sum_{\underline{i}\in I} c_{\underline{i}}P_{\underline{i}}$$

satisfy

$$1 = \sum_{\underline{i}\in I} c_{\underline{i}}\,.$$

Proof. We note that $I \neq \emptyset$ implies $\underline{0} \in I$. Now $P_{\underline{0}} = Q_{\underline{0}}$ and clearly

$$P_{\underline{0}} P_I = Q_{\underline{0}} P_I = \sum_{\underline{i} \in I} Q_{\underline{0}} Q_{\underline{i}} = Q_{\underline{0}} = P_{\underline{0}} .$$

Now we also have that $P_{\underline{0}} P_{\underline{i}} = P_{\underline{0}}$ for all $\underline{i} \in \mathbb{N}^d$ and therefore

$$P_{\underline{0}} = P_{\underline{0}} P_I = P_{\underline{0}} \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}} = P_{\underline{0}} \sum_{\underline{i} \in I} c_{\underline{i}} ,$$

from which the desired result follows (noting that for each k one has $V_0^k \neq \{0\}$). □

We have spent some time now building up these adaptive sparse grids but we should check that the classical combination technique comes out of this.

Lemma 4.25. *Let $I = \{\underline{i} \in \mathbb{N}^d : |\underline{i}| \leq n\}$, then the $c_{\underline{i}}$ corresponding to P_I satisfy*

$$c_{\underline{i}} = (-1)^{n-|\underline{i}|} \binom{d-1}{n-|\underline{i}|} .$$

Proof. We know from Proposition 4.21 that

$$c_{\underline{i}} = \sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} (-1)^{|\underline{j}-\underline{i}|} \chi_I(\underline{j}) ,$$

therefore for $I = \{\underline{i} \in \mathbb{N}^d : |\underline{i}| \leq n\}$ and $\underline{i}$ such that $|\underline{i}| = n-k$ for some $k \in \{0, \dots, d-1\}$ one has

$$c_{\underline{i}} = \sum_{l=0}^{k} (-1)^l \binom{d}{l} .$$

With an induction argument on k using the identity $\binom{d}{k} - \binom{d-1}{k-1} = \binom{d-1}{k}$ it is easily shown that

$$c_{\underline{i}} = (-1)^k \binom{d-1}{k} .$$

Substituting $k = n - |\underline{i}|$ and recognising that $c_{\underline{i}} = 0 =: \binom{d-1}{n-|\underline{i}|}$ for $|\underline{i}| \leq n-d$ and $|\underline{i}| > n$ completes the proof. □

For the next proposition we restrict ourselves to considering nested spaces of piecewise multi-linear functions so that we may formulate an error bound for interpolation onto adaptive sparse grids. Let $V = C([0,1]^d)$, that is the space of bounded continuous functions on $[0,1]^d$. Further, for $\underline{i} \in \mathbb{N}^d$ we define $V_{\underline{i}}$ to be the space of piecewise multi-linear functions which interpolate between function values given on the grid $\Omega_{\underline{i}} = \{x_{\underline{i},\underline{j}} = (j_1 2^{-i_1}, \dots, j_d 2^{-i_d}) : \underline{0} \le \underline{j} \le 2^{\underline{i}} = (2^{i_1}, \dots, 2^{i_d})\}$, that is $V_{\underline{i}} = \mathrm{span}\{\phi_{\underline{i},\underline{j}} : \underline{0} \le \underline{j} \le 2^{\underline{i}}\}$ with the nodal basis functions $\phi_{\underline{i},\underline{j}}$ as in Definition 2.10. We consider the projections $P_{\underline{i}} : V \to V_{\underline{i}}$ given by the Lagrange interpolation formula

$$P_{\underline{i}} u = \sum_{\underline{0} \le \underline{j} \le 2^{\underline{i}}} u(x_{\underline{i},\underline{j}}) \phi_{\underline{i},\underline{j}} \,, \tag{4.2}$$

where $u \in V$. Given a downset $I \subset \mathbb{N}^d$ one obtains projections $P_I : V \mapsto V_I = \sum_{\underline{i} \in I} V_{\underline{i}}$ via the combination

$$P_I = \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}} \,.$$

We will use the notation $u_{\underline{i}} := P_{\underline{i}} u$ and $u_I := P_I u$ leading to the general combination formula

$$u_I = \sum_{\underline{i} \in I} c_{\underline{i}} u_{\underline{i}} \,, \tag{4.3}$$

with the coefficients determined by Proposition 4.21. If the $P_{\underline{i}} u$ for $u \in V$ are defined to be the piecewise linear interpolants of $u(\Omega_{\underline{i}})$ then the u_I are simply the sparse grid interpolants of u onto V_I. This leads us to the following error estimate.

Proposition 4.26. *Let $I \subset \mathbb{N}^d_+$ be a downset, $u \in H^2_{0,mix}$ and $V_{\underline{i}}$, $P_{\underline{i}}$, $u_{\underline{i}}$ as in* (4.2)*. Let $c_{\underline{i}} \in \mathbb{R}$ be the combination coefficients*

$$P_I = \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}} \,,$$

and let u_I be as given in (4.3)*. Then*

$$\|u - u_I\|_2 \le 3^{-d} \|D^{\underline{2}} u\|_2 \left(3^{-d} - \sum_{\underline{1} \le \underline{i} \in I} 2^{-2|\underline{i}|} \right) . \tag{4.4}$$

Proof. We note that as u is zero on the boundary one has

$$u_{\underline{i}} = \sum_{\underline{1} \le \underline{i}} u^h_{\underline{i}} \quad \text{and} \quad u_I = \sum_{\underline{1} \le \underline{i} \in I} u^h_{\underline{i}}$$

where $u^h_{\underline{i}} \in W_{\underline{i}}$ are the hierarchical surplus' of u (where $W_{\underline{i}} = Q_{\underline{i}}$ is defined in Section 2.1). It follows that

$$\|u - u_I\|_2 \le \| \sum_{\underline{1} \le \underline{i} \notin I} u^h_{\underline{i}} \|_2 \le \sum_{\underline{1} \le \underline{i} \notin I} \|u^h_{\underline{i}}\|_2 \,.$$

By applying Lemma 2.15 we have

$$\|u - u_I\|_2 \leq 3^{-d} \|D^{\underline{2}} u\|_2 \sum_{\underline{1} \leq \underline{i} \notin I} 2^{-2|\underline{i}|} \,.$$

Using the fact that

$$\sum_{\underline{1} \leq \underline{i} \in I} 2^{-2|\underline{i}|} + \sum_{\underline{1} \leq \underline{i} \notin I} 2^{-2|\underline{i}|} = \sum_{\underline{1} \leq \underline{i}} 2^{-2|\underline{i}|} = \sum_{i_1,\dots,i_d=1}^{\infty} 2^{-2i_1} \cdots 2^{-2i_d} = 3^{-d}$$

one obtains the desired result. □

This can be extended to a similar result on H^2_{mix} in a the same manner as the extension of the classical sparse grids in Section 2.1. Because the classical combination technique interpolates exactly to the sparse grid and also performs well for more general problems we expect the same to be true of these adaptive sparse grids. We will provide a general error splitting estimate but first we require a lemma.

Wong [127] shows that a combination projected onto a subset of its dimensions results in a valid combination on these dimensions. We can provide an alternative proof within this framework. Let $I \subset \mathbb{N}^d$ be a downset, $1 \leq k \leq d$ and $\{e_1, \dots, e_k\} \subset \{1, \dots, d\}$. We define

$$I_{e_1,\dots,e_k} = \{\underline{i} \in \mathbb{N}^k : \underline{i} = (j_{e_1}, \dots, j_{e_k}) \text{ for some } \underline{j} \in I\} \,,$$

and for $\underline{l} \in \mathbb{N}^k$ we also define

$$I_{\underline{l}|e_1,\dots,e_k} = \{\underline{i} \in I : (i_{e_1}, \dots, i_{e_k}) = (l_1, \dots, l_k)\}.$$

These two definitions allow us to write

$$\sum_{\underline{i} \in I} f(\underline{i}) = \sum_{\underline{l} \in I_{e_1,\dots,e_k}} \left(\sum_{\underline{i} \in I_{\underline{l}|e_1,\dots,e_k}} f(\underline{i}) \right) . \tag{4.5}$$

Clearly given $I \in \mathcal{D}(\mathbb{N}^d)$ then $I_{e_1,\dots,e_k}$ is a downset in $\mathbb{N}^k$. We have the following lemma.

Lemma 4.27. *Let $I \in \mathcal{D}(\mathbb{N}^d)$ and $\{c_{\underline{i}}\}_{\underline{i} \in I}$ be coefficients corresponding to the projection $P_I = \sum_{\underline{i} \in I} c_{\underline{i}} P_{\underline{i}}$. Further, fix $1 \leq k \leq d$ and $\{e_1, \dots, e_k\} \subset \{1, \dots, d\}$ and let*

$$P_{I_{e_1,\dots,e_k}} : V^{e_1} \otimes \cdots \otimes V^{e_k} \to V_{I_{e_1,\dots,e_k}} \,,$$

where $V_{I_{e_1,\dots,e_k}} := \sum_{\underline{i}\in I_{e_1,\dots,e_k}} V^{e_1}_{i_1}\otimes\cdots\otimes V^{e_k}_{i_k}$*. Additionally let* $\{\bar{c}_{\underline{j}}\}_{\underline{j}\in I_{e_1,\dots,e_k}}$ *be coefficients corresponding to the projection* $P_{I_{e_1,\dots,e_k}} = \sum_{\underline{j}\in I_{e_1,\dots,e_k}} \bar{c}_{\underline{j}} \bigotimes_{l=1}^k P^{e_l}_{j_l}$*, then for all* $\underline{j}\in I_{e_1,\dots,e_k}$ *one has*

$$\bar{c}_{\underline{j}} = \sum_{\underline{i}\in I_{\underline{j}|e_1,\dots,e_k}} c_{\underline{i}}\,.$$

Proof. Consider a function $u \in V = V^1\otimes\cdots\otimes V^d$ which only depends on the coordinates $x_{e_1},\dots,x_{e_k}$, that is $u(x_1,\dots,x_d) = v(x_{e_1},\dots,x_{e_k})$ for some $v\in V^{e_1}\otimes\cdots\otimes V^{e_k}$. It follows that $P_{\underline{i}}u = \bigotimes_{l=1}^k P^{e_l}_{i_{e_l}} v$ for all $\underline{i}\in I$ and therefore

$$\begin{aligned}
P_I u = \sum_{\underline{i}\in I} c_{\underline{i}} P_{\underline{i}} u &= \sum_{\underline{i}\in I} c_{\underline{i}} \bigotimes_{l=1}^k P^{e_l}_{i_{e_l}} v \\
&= \sum_{\underline{j}\in I_{e_1,\dots,e_k}} \sum_{\underline{i}\in I_{\underline{j}|e_1,\dots,e_d}} c_{\underline{i}} \left(\bigotimes_{l=1}^k P^{e_l}_{i_{e_l}} v \right) \\
&= \sum_{\underline{j}\in I_{e_1,\dots,e_k}} \left(\bigotimes_{l=1}^k P^{e_l}_{j_l} v \right) \sum_{\underline{i}\in I_{\underline{j}|e_1,\dots,e_d}} c_{\underline{i}}\,.
\end{aligned}$$

Finally, since it is clear that $P_I u = P_{I_{e_1,\dots,e_k}} v$ and u depending on only $x_{e_1},\dots,x_{e_k}$ was arbitrary, one has the desired result. □

This result allows us to write down a general formula regarding error estimates of dimension adaptive sparse grids when an error splitting is assumed. Here we show that the coefficients obtained via the projection framework produce good results when applied to a larger class of problems, namely those which satisfy an error splitting.

Proposition 4.28. *Let* $I \in \mathcal{D}(\mathbb{N}^d)$ *be non-empty. For each* $1\le k\le d$ *and* $\{e_1,\dots,e_k\}\subset\{1,\dots,d\}$*, we define* $c_{I_{e_1,\dots,e_k},\underline{j}}$ *for* $\underline{j}\in I_{e_1,\dots,e_k}$ *to be the coefficients corresponding to the projection*

$$P_{I_{e_1,\dots,e_k}} = \sum_{\underline{j}\in I_{e_1,\dots,e_k}} c_{I_{e_1,\dots,e_k},\underline{j}} \bigotimes_{l=1}^k P^{e_l}_{j_l}\,.$$

Consider the corresponding combination

$$u_I := \sum_{\underline{i}\in I} c_{I,\underline{i}} u_{\underline{i}}$$

with each $u_{\underline{i}}$ *being an approximation of a function* u *which satisfies*

$$u - u_{\underline{i}} = \sum_{k=1}^d \sum_{\{e_1,\dots,e_k\}\subset\{1,\dots,d\}} C_{e_1,\dots,e_k}(h_{i_{e_1}},\dots,h_{i_{e_k}}) h^{p_{e_1}}_{i_{e_1}}\cdots h^{p_{e_k}}_{i_{e_k}}\,, \tag{4.6}$$

where $p_1, \dots, p_d > 0$ and for each $\{e_1, \dots, e_k\} \subset \{1, \dots, d\}$ one there exists $K_{e_1,\dots,e_k} > 0$ such that $|C_{e_1,\dots,e_k}(h_{i_{e_1}}, \dots, h_{i_{e_k}})| \leq K_{e_1,\dots,e_k}$ for all $h_{i_{e_1}}, \dots, h_{i_{e_k}}$. Then

$$|u - u_I| \leq \sum_{k=1}^{d} \sum_{\{e_1,\dots,e_k\}\subset\{1,\dots,d\}} K_{e_1,\dots,e_k} \sum_{\underline{j}\in I_{e_1,\dots,e_k}} |c_{I_{e_1,\dots,e_k},\underline{j}}| h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} .$$

Proof. Since $\sum_{\underline{i}\in I} c_{I,\underline{i}} = 1$ by Lemma 4.24 we have

$$u - u_I = \sum_{\underline{i}\in I} c_{I,\underline{i}} (u - u_{\underline{i}}) .$$

From here we substitutes the error splitting formula (4.6) and take the absolute value of both sides to obtain

$$\begin{aligned} |u - u_I| &= \left| \sum_{\underline{i}\in I} c_{I,\underline{i}} \sum_{k=1}^{d} \sum_{\{e_1,\dots,e_k\}\subset\{1,\dots,d\}} C_{e_1,\dots,e_k}(h_{i_{e_1}}, \dots, h_{i_{e_k}}) h_{i_{e_1}}^{p_{e_1}} \cdots h_{i_{e_k}}^{p_{e_k}} \right| \\ &\leq \sum_{k=1}^{d} \sum_{\{e_1,\dots,e_k\}\subset\{1,\dots,d\}} K_{e_1,\dots,e_k} \left| \sum_{\underline{i}\in I} c_{I,\underline{i}} h_{i_{e_1}}^{p_{e_1}} \cdots h_{i_{e_k}}^{p_{e_k}} \right| . \end{aligned}$$

Now the inner most sum we can rewrite as

$$\begin{aligned} \sum_{\underline{i}\in I} c_{I,\underline{i}} h_{i_{e_1}}^{p_{e_1}} \cdots h_{i_{e_k}}^{p_{e_k}} &= \sum_{\underline{j}\in I_{e_1,\dots,e_k}} \sum_{\underline{i}\in I_{\underline{j}|e_1,\dots,e_k}} c_{I,\underline{i}} h_{i_{e_1}}^{p_{e_1}} \cdots h_{i_{e_k}}^{p_{e_k}} \\ &= \sum_{\underline{j}\in I_{e_1,\dots,e_k}} h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} \sum_{\underline{i}\in I_{\underline{j}|e_1,\dots,e_k}} c_{I,\underline{i}} . \end{aligned}$$

Using Lemma 4.27 we have $c_{I_{e_1,\dots,e_k},\underline{j}} = \sum_{\underline{i}\in I_{\underline{j}|e_1,\dots,e_k}} c_{I,\underline{i}}$. Substituting these back into the inequality for $|u - u_I|$ we have

$$\begin{aligned} |u - u_I| &\leq \sum_{k=1}^{d} \sum_{\{e_1,\dots,e_k\}\subset\{1,\dots,d\}} K_{e_1,\dots,e_k} \left| \sum_{\underline{j}\in I_{e_1,\dots,e_k}} h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} c_{I_{e_1,\dots,e_k},\underline{j}} \right| \\ &\leq \sum_{k=1}^{d} \sum_{\{e_1,\dots,e_k\}\subset\{1,\dots,d\}} K_{e_1,\dots,e_k} \sum_{\underline{j}\in I_{e_1,\dots,e_k}} |c_{I_{e_1,\dots,e_k},\underline{j}}| h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} , \end{aligned}$$

which is the desired result. □

Whilst this result is not simple enough that one could easily write down the resulting bound for a given I, it does give one a way to quickly compute the bound.

4.3 Multi-variate Extrapolation

In this section we study extrapolation methods in the context of sparse grids. First we give a brief review of extrapolation methods, in particular the multi-variate/splitting extrapolation methods which have been studied extensively in the literature, see for example [115, 17, 89]. We look at how multi-variate extrapolation can be viewed as a sparse grid approximation to classical Richardson extrapolation and through some examples demonstrate how the coefficients of other truncated combinations can be modified to achieve a similar effect. Second, we will review Reisinger's approach of applying the classical combination technique to $\tilde{u}_{\underline{i}}$ which are extrapolations [109]. We show how these combinations can be studied within the framework of adaptive sparse grids developed in Section 4.2. This leads to an adaptive extrapolation method for which the combination coefficients are easily computed.

Consider approximations $u_{\underline{i}}$ for $\underline{i} \in \mathbb{N}^d$ which satisfy an extended version of the error splitting model (2.22) defined as follows.

Definition 4.29. Given integers $0 < p < q$ an approximation $u_{\underline{i}} : \Omega \subset \mathbb{R}^d \mapsto \mathbb{R}$ of $u : \Omega \mapsto \mathbb{R}$ is said to satisfy the *p, q extended error splitting model* if

$$u - u_{\underline{i}} = \sum_{k=1}^{d} \sum_{\{j_1,\dots,j_k\} \subset \{1,\dots,d\}} C_{j_1,\dots,j_k} h_{i_{j_1}}^p \cdots h_{i_{j_k}}^p + R_{\underline{i}} \tag{4.7}$$

where each $C_{j_1,\dots,j_k}$ term depends only upon $\boldsymbol{x} \in \Omega$, and the remainder term $R_{\underline{i}}$ has the form

$$R_{\underline{i}} = \sum_{k=1}^{d} \sum_{\{j_1,\dots,j_k\} \subset \{1,\dots,d\}} D_{j_1,\dots,j_k}(h_{i_{j_1}}, \dots, h_{i_{j_d}}) h_{i_{j_1}}^q \cdots h_{i_{j_k}}^q \tag{4.8}$$

and there exists finite $K > 0$ such that $|D_{j_1,\dots,j_k}| \leq K$ for all $k \in \{1, \dots, d\}$ and $\{j_1, \dots, j_k\} \subset \{1, \dots, d\}$.

This was the error model used in the work of Reisinger [109]. The motivation behind this error model is that as the $C_{j_1,\dots,j_k}$ do not depend on $h_{i_1}, \dots, h_{i_d}$ which makes it possible to cancel these terms out by adding approximations $u_{\underline{i}}$ for different $\underline{i}$ given the right coefficients. Notice that the $R_{\underline{i}}$ term is equivalent to the previous error splitting model (2.22) but with exponent $q > p$. This error splitting model can typically be shown for a given problem and discretisation in the same way one derives the error model (2.22) with the exception that additional terms in the Taylor series expansion are retained.

4.3.1 Richardson and Multi-Variate Extrapolation

Classical Richardson extrapolation [111] involves taking the two approximations $u_{\underline{i}}$ and $u_{\underline{i}+\underline{1}}$ of order $p > 0$ and combining them according to the formula

$$u_{\underline{i}}^{re} := \frac{2^p}{2^p-1}u_{\underline{i}+\underline{1}} - \frac{1}{2^p-1}u_{\underline{i}} \tag{4.9}$$

in order to obtain a higher order approximation of u. For functions with only one spatial dimension satisfying the error model of the form $u - u_j = Ch_j^p + \mathcal{O}(h_j^q)$ for $j \in \mathbb{N}$ and $0 < p < q$ one has

$$u - u_j^{re} = \frac{2^p(u-u_{j+1})}{2^p-1} - \frac{u-u_j}{2^p-1} = \mathcal{O}(h_j^q)\,,$$

and thus one achieves an order q approximation rather than p. For functions in higher dimensions error models typically have many more terms, not all of which will cancel. For example, with the p, q extended error splitting model only the $C_1, \dots, C_d$ terms are eliminated by this extrapolation as is shown in the following result.

Proposition 4.30. *Let $0 < p < q$ and $u_{\underline{i}}$, $u_{\underline{i}+\underline{1}}$ be approximations of u satisfying the p, q extended error splitting model, then the approximation $u_{\underline{i}}^{re}$ defined by (4.9) satisfies*

$$\begin{aligned}\left|u - u_{\underline{i}}^{re}\right| \leq &\sum_{k=2}^{d} \sum_{\{j_1,\dots,j_k\}\subset\{1,\dots,d\}} \frac{1-2^{-(k-1)p}}{2^p-1}|C_{j_1,\dots,j_k}|h_{i_{j_1}}^p \cdots h_{i_{j_k}}^p \\ &+ \sum_{k=1}^{d} \sum_{\{j_1,\dots,j_k\}\subset\{1,\dots,d\}} \frac{2^{p-kq}+1}{2^p-1} K h_{i_{j_1}}^q \cdots h_{i_{j_k}}^q\,.\end{aligned}$$

Observe that the sum over the $C_{j_1,\dots,j_k}$ terms starts at $k = 2$ as the $k = 1$ terms have been eliminated.

Proof. We start by noting that

$$u - u_{\underline{i}}^{re} = \frac{2^p}{2^p-1}(u - u_{\underline{i}+\underline{1}}) - \frac{1}{2^p-1}(u - u_{\underline{i}})\,.$$

Substituting (4.7) then collecting each of the $C_{j_1,\dots,j_k}$ terms we observe that

$$\frac{2^p}{2^p-1}C_{j_1,\dots,j_k}h_{i_{j_1}+1}^p \cdots h_{i_{j_k}+1}^p - \frac{1}{2^p-1}C_{j_1,\dots,j_k}h_{i_{j_1}}^p \cdots h_{i_{j_k}}^p = \frac{2^{-(k-1)p}-1}{2^p-1}C_{j_1,\dots,j_k}h_{i_{j_1}}^p \cdots h_{i_{j_k}}^p\,,$$

for which the right hand side is 0 when $k = 1$. Thus we obtain

$$\begin{aligned}\left|u - u_{\underline{i}}^{re}\right| \le &\sum_{k=2}^{d} \sum_{\{j_1,\dots,j_k\}\subset\{1,\dots,d\}} \frac{1-2^{-(k-1)p}}{2^p-1}|C_{j_1,\dots,j_k}|h_{i_{j_1}}^p \cdots h_{i_{j_k}}^p \\ &+ \left|\frac{2^pR_{\underline{i}+\underline{1}} - R_{\underline{i}}}{2^p-1}\right| .\end{aligned}$$

For the $D_{j_1,\dots,j_k}$ terms in $R_{\underline{i}}$ we observe that

$$\begin{aligned}&\left|\frac{2^p}{2^p-1}D_{j_1,\dots,j_k}(h_{i_{j_1}+1},\dots,h_{i_{j_k}+1})h_{i_{j_1}+1}^q \cdots h_{i_{j_k}+1}^q\right. \\ &\qquad \left. -\frac{1}{2^p-1}D_{j_1,\dots,j_k}(h_{i_{j_1}},\dots,h_{i_{j_k}})h_{i_{j_1}}^q \cdots h_{i_{j_k}}^q\right| \le \frac{2^{p-kq}+1}{2^p-1}Kh_{i_{j_1}}^q \cdots h_{i_{j_k}}^q ,\end{aligned}$$

which leads to the desired result. □

From this result it may seem that Richardson extrapolation is not as effective for functions in higher dimensions. However, if we assume that our computational grids are isotropic then the following result is obtained.

Corollary 4.31. *Let $0 < p < q$, $\underline{i} = (n,\dots,n)$ for some $n \in \mathbb{N}$, and $u_{\underline{i}}$, $u_{\underline{i}+\underline{1}}$ be approximations of u satisfying the p,q extended error splitting model, then*

$$\left|u - \frac{2^pu_{\underline{i}+\underline{1}} - u_{\underline{i}}}{2^p-1}\right| = \mathcal{O}(2^{-n\min\{2p,q\}}) .$$

Proof. As $h_{i_k} = 2^{-n}$ for each $k \in \{1,\dots,d\}$ one obtains

$$\begin{aligned}\left|u - u_{\underline{i}}^{re}\right| \le &\sum_{k=2}^{d} \sum_{\{j_1,\dots,j_k\}\subset\{1,\dots,d\}} \frac{1-2^{-(k-1)p}}{2^p-1}|C_{j_1,\dots,j_k}|2^{-npk} \\ &+ \sum_{k=1}^{d} \sum_{\{j_1,\dots,j_k\}\subset\{1,\dots,d\}} \frac{2^{p-kq}+1}{2^p-1}K2^{-npq} .\end{aligned}$$

As the dominating terms are those for $k = 2$ in the first sum and $k = 1$ in the second sum one obtains the desired asymptotic result. □

Therefore we see that if the computational grids are isotropic the order of convergence is $\min\{2p,q\}$ which is better than the order p convergence obtained for $|u - u_{\underline{i}}|$

Remark 4.32. There is a difficulty with using Richardson extrapolation with finite difference methods. Consider the domain $\Omega = [0,1]^d$ discretised by the collection of grids $\Omega_{\underline{i}} = \{(j_1 2^{-i_1}, \dots, j_d 2^{-i_d}) : \underline{0} \le \underline{j} \le 2^{\underline{i}}\}$ for $\underline{i} \in \mathbb{N}^d$ (and $2^{\underline{i}} := (2^{i_1}, \dots, 2^{i_d})$). Approximations $u_{\underline{i}+\underline{1}}$ and $u_{\underline{i}}$ consist of function values stored on the grids $\Omega_{\underline{i}+\underline{1}}$ and $\Omega_{\underline{i}}$ respectively for which the error splitting model is satisfied at these grid points. To add the approximations they must first be represented on a common grid. This means interpolating $u_{\underline{i}}$ onto $\Omega_{\underline{i}+\underline{1}}$ or down-sampling $u_{\underline{i}+\underline{1}}$ to $\Omega_{\underline{i}}$. For the former, we require the solution to be sufficiently smooth and the interpolation to have order more than p or else the constants $C_{j_1,\dots,j_k}$ will potentially be different on the grid points $\Omega_{\underline{i}+\underline{1}} \backslash \Omega_{\underline{i}}$ and the extrapolation could give very different results. Down-sampling $u_{\underline{i}+\underline{1}}$ to $\Omega_{\underline{i}}$ is straightforward and has almost negligible cost compared to interpolation. As a result, down-sampling is typically the preferred option.

Richardson extrapolation is somewhat expensive compared to the computation of $u_{\underline{i}}$ alone. If the computation time is proportional to the number of unknowns that it is clear that the computation of (4.9) costs approximately $2^d + 1$ times that of $u_{\underline{i}}$. The question we now ask is if we can obtain a similar result for smaller cost.

We now consider the so called multi-variate extrapolation formula

$$u_{\underline{i}}^{mve} := \frac{2^p}{2^p - 1} \sum_{m=1}^{d} u_{\underline{i}+\underline{e}_m} - \frac{1 + (d-1)2^p}{2^p - 1} u_{\underline{i}} \,, \tag{4.10}$$

where $\underline{e}_m$ is the multi-index which is 1 in the mth term and 0 elsewhere. For multi-variate extrapolation we have the following result.

Proposition 4.33. *Let $0 < p < q$ and $u_{\underline{i}}$, $u_{\underline{i}+\underline{e}_m}$ for $m \in \{1, \dots, d\}$ be approximations of u satisfying the p, q extended error splitting model, then the approximation $u_{\underline{i}}^{mve}$ defined by* (4.10) *satisfies*

$$\begin{aligned}\left|u - u_{\underline{i}}^{mve}\right| \le & \sum_{k=2}^{d} \sum_{\{j_1,\dots,j_k\} \subset \{1,\dots,d\}} (k-1) |C_{j_1,\dots,j_k}| h_{i_{j_1}}^p \cdots h_{i_{j_k}}^p \\ & + \sum_{k=1}^{d} \sum_{\{j_1,\dots,j_k\} \subset \{1,\dots,d\}} \left(\frac{2d2^p}{2^p - 1} + 2d - k - 1\right) K h_{i_{j_1}}^q \cdots h_{i_{j_k}}^q \,.\end{aligned}$$

Proof. As the coefficients sum to one, that is

$$d\frac{2^p}{2^p - 1} - \frac{1 + (d-1)2^p}{2^p - 1} = \frac{2^p - 1}{2^p - 1} = 1 \,,$$

we have

$$u - u_{\underline{i}}^{mve} = \frac{2^p}{2^p - 1} \sum_{m=1}^{d} (u - u_{\underline{i}+\underline{e}_m}) - \frac{1 + (d-1)2^p}{2^p - 1} (u - u_{\underline{i}}) \,.$$

We now need only substitute the error splitting model and look at what happens to each term. As before we see that for $k = 1$ the C_{j_1} terms (for each $j_1 \in \{1,\dots,d\}$) cancel as

$$\frac{2^p}{2^p-1}\left(C_{j_1}h^p_{i_{j_1}+1} + (d-1)C_{j_1}h^p_{i_{j_1}}\right) - \frac{1+(d-1)2^p}{2^p-1}C_{j_1}h^p_{i_{j_1}} = 0\,.$$

For $k = 2,\dots,d$ and $\{j_1,\dots,j_k\} \subset \{1,\dots,d\}$ we obtain

$$\begin{aligned}\frac{C_{j_1,\dots,j_k}}{2^p-1}h^p_{i_{j_1}}\cdots h^p_{i_{j_k}}\left(2^p\left(k2^{-p}+(d-k)\right)-(1+(d-1)2^p)\right)&\\ = (1-k)C_{j_1,\dots,j_k}h^p_{i_{j_1}}\cdots h^p_{i_{j_k}}\,.&\end{aligned}$$

Therefore we obtain

$$\begin{aligned}\left|u-u^{mve}_{\underline{i}}\right| \le &\sum_{k=2}^{d}\sum_{\{j_1,\dots,j_k\}\subset\{1,\dots,d\}}(k-1)|C_{j_1,\dots,j_k}|h^p_{i_{j_1}}\cdots h^p_{i_{j_k}}\\ &+\left|\frac{2^p}{2^p-1}\sum_{m=1}^{d}R_{\underline{i}+\underline{e}_m}-\frac{1+(d-1)2^p}{2^p-1}R_{\underline{i}}\right|\,.\end{aligned}$$

For $k \in \{1,\dots,d\}$ each $D_{j_1,\dots,j_k}$ term in the $R_{\underline{i}}$ satisfies

$$\begin{aligned}\Bigg|\frac{2^p}{2^p-1}\sum_{m=1}^{k}D_{j_1,\dots,j_k}(h_{i_{j_1}+\delta_{j_1,m}},\dots,h_{i_{j_k}+\delta_{j_k,m}})h^q_{i_{j_1}+\delta_{j_1,m}}\cdots h^q_{i_{j_k}+\delta_{j_k,m}}&\\ -\frac{1+(d-1)2^p}{2^p-1}D_{j_1,\dots,j_k}(h_{i_{j_1}},\dots,h_{i_{j_k}})h^q_{i_{j_1}}\cdots h^q_{i_{j_k}}\Bigg|&\\ \le\frac{k2^{p-q}+1+(2d-k-1)2^p}{2^p-1}Kh^q_{i_{j_1}}\cdots h^q_{i_{j_k}}\,.&\end{aligned}$$

Now as

$$\frac{k2^{p-q}+1+(2d-k-1)2^p}{2^p-1} \le \frac{2d2^p}{2^p-1}+2d-k-1$$

we obtain the desired result. □

Thus we observe that the multi-variate extrapolation results in cancellation of the same terms as the classical Richardson extrapolation. Further, if $\underline{i}$ is assumed to correspond to an isotropic grid (that is $i_1 = i_1 = \cdots = i_d$) we obtain a result similar to that of Corollary 4.31 If the cost of computing the $u_{\underline{i}}$ is proportional to the number of unknowns then the cost of computing $u^{mve}_{\underline{i}}$ is approximately $2d+1$ times that of computing $u_{\underline{i}}$ which for large d is significantly smaller than the cost of Richardson extrapolation ($\approx 2^d + 1$ times cost of $u_{\underline{i}}$). The trade-off is that the constants in the error bound for $u_{\underline{i}}^{mve}$ are larger than those obtained for Richardson extrapolation.

There is a nice connection between the multi-variate extrapolation formula and the truncated combination technique. Consider the truncated combination

$$u_{1,\underline{i}}^t = \sum_{m=1}^{d} u_{\underline{i}+\underline{e}_m} - (d-1)u_{\underline{i}} \,.$$

The full grid corresponding to this truncated combination is $u_{\underline{f}}$ where $\underline{f} = \underline{i} + \underline{1}$. Therefore we could consider $u_{1,\underline{i}}^t$ to be an approximation of $u_{\underline{i}+\underline{1}}$. If we make this substitution in the classical Richardson extrapolation (4.9) then one obtains

$$\frac{2^p}{2^p-1}\left(\sum_{m=1}^{d} u_{\underline{i}+\underline{e}_m} - (d-1)u_{\underline{i}}\right) - \frac{1}{2^p-1}u_{\underline{i}}$$

which is precisely the multi-variate extrapolation (4.10) once the $u_{\underline{i}}$ terms are collected. Thus the multi-variate extrapolation can be viewed as a sparse grid approximation of the classical Richardson extrapolation.

This observation can be applied to develop additional extrapolation formula. First observe that the classical Richardson extrapolation formula can be extended to

$$\frac{2^{sp}}{2^{sp}-1}u_{\underline{i}+\underline{s}} - \frac{1}{2^{sp}-1}u_{\underline{i}} \,.$$

for some $s \in \mathbb{N}_+$ and $\underline{s} = (s, \dots, s)$. It is straightforward to show that this results in the same cancellation of error terms as (4.9). The cost of computing this is approximately $2^{sd}+1$ times that of $u_{\underline{i}}$ where the cost is proportional to the unknowns. However, if we are to replace $u_{\underline{i}+\underline{s}}$ with the truncated combination $u_{\underline{s}}^t$ then we expect to obtain a similar result but with a cost which is proportional to $(s+d-1)^{d-1}2^s$ times that of $u_{\underline{i}}$ for large s. The fact that the C_{j_1} terms still cancel for $j_1 \in \{1, \dots, d\}$ comes from the fact that the truncated combination over these terms produces $C_{j_1} h_{i_{j_1}+s}^p$ which is identical to these terms in the error splitting for $u_{\underline{i}+\underline{s}}$. This means that when computing a truncated combination $u_{s,\underline{i}}^t$ we can attempt to obtain a higher order approximation by computing $u_{\underline{i}}$ in addition to the coarse solutions required for $u_{s,\underline{i}}^t$. The difficulty with this however is that for large s, downsampling the solution to $\Omega_{\underline{i}}$ means many of the high frequency components from $u_{s,\underline{i}}^t$ are lost. One would need to determine whether higher accuracy function values on $\Omega_{\underline{i}}$ outweighs the contribution from higher frequencies. The alternative is to interpolate all coarse solutions to the sparse (or full) grid, but requires interpolation of order more than p for the extrapolation to be effective.

4.3.2 Combinations of extrapolations

Reisinger [109] considered a different extrapolation formula which cancels all of the $C_{j_1,\dots,j_k}$ terms in the extended error splitting model leaving only contributions from $R_{\underline{i}}$. He then considered replacing each of the $u_{\underline{i}}$ in the classical combination technique with extrapolations of this kind. We review this work and then analyse it from the perspective of adaptive sparse grids. This will enable us to give compact formula for computing the combination coefficients and also leads to and adaptive extrapolation algorithm.

For $u_{\underline{i}}$ satisfying the p, q extended error splitting model with $p = 2$ and $q > p$ we consider the extrapolation formula

$$\tilde{u}_{\underline{i}} := \sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} \frac{(-4)^{|\underline{j}-\underline{i}|}}{(-3)^d} u_{\underline{j}} \,. \tag{4.11}$$

To show why it is this works as an extrapolation formula we first have a Lemma.

Lemma 4.34 ([109]). *One has*

$$\sum_{\underline{0} \leq \underline{j} \leq \underline{1}} \frac{(-4)^{|\underline{j}|}}{(-3)^d} = 1 \,.$$

Proof.

$$\sum_{\underline{0} \leq \underline{j} \leq \underline{1}} \frac{(-4)^{|\underline{j}|}}{(-3)^d} = 3^{-d} \left(\sum_{j_1=0}^{1} (-4)^{j_1} \right) \cdots \left(\sum_{j_d=0}^{1} (-4)^{j_d} \right) = (-3)^{-d} \prod_{k=1}^{d} (1-4) = 1 \,.$$

□

Now we can analyse what $\tilde{u}_{\underline{i}}$ looks like when the $u_{\underline{i}}$ satisfy the extended error splitting model.

Proposition 4.35 ([109]). *Let $u_{\underline{i}}$ satisfy the p, q extended error splitting model for $q > p = 2$, then*

$$|u - \tilde{u}_{\underline{i}}| \leq \frac{5^d}{3^d} K \sum_{k=1}^{d} \sum_{\{l_1,\dots,l_k\} \subset \{1,\dots,d\}} \left(\frac{1 + 2^{2-q}}{5} \right)^k h_{i_{l_1}}^q \cdots h_{i_{l_k}}^q \,.$$

Proof. Lemma 4.34 tells us that

$$\begin{aligned} u - \tilde{u}_{\underline{i}} &= \sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} \frac{(-4)^{|\underline{j}-\underline{i}|}}{(-3)^d} (u - u_{\underline{j}}) \\ &= \sum_{\underline{0} \leq \underline{j} \leq \underline{1}} \frac{(-4)^{|\underline{j}|}}{(-3)^d} (u - u_{\underline{i}+\underline{j}}) \,. \end{aligned}$$

From here we can substitute in the error splitting model (4.7) to obtain

$$u - \tilde{u}_{\underline{i}} = \sum_{\underline{0} \leq \underline{j} \leq \underline{1}} \frac{(-4)^{|\underline{j}|}}{(-3)^d} \left(\sum_{k=1}^{d} \sum_{\{l_1,\dots,l_k\} \subset \{1,\dots,d\}} C_{l_1,\dots,l_k} h^2_{i_{l_1}+j_{l_1}} \cdots h^2_{i_{l_k}+j_{l_k}} + R_{\underline{i}+\underline{j}} \right) .$$

We look at an arbitrary $C_{l_1,\dots,l_k}$ term, hence fix $k \in \{1,\dots,d\}$ and $\{l_1,\dots,l_k\} \subset \{1,\dots,d\}$, then we have

$$\sum_{\underline{0} \leq \underline{j} \leq \underline{1}} \frac{(-4)^{|\underline{j}|}}{(-3)^d} C_{l_1,\dots,l_k} h^2_{j_{l_1}+i_{l_1}} \cdots h^2_{j_{l_k}+i_{l_k}} = \frac{C_{l_1,\dots,l_k}}{(-3)^d} h^2_{i_{l_1}} \cdots h^2_{i_{l_k}} \sum_{\underline{0} \leq \underline{j} \leq \underline{1}} (-4)^{|\underline{j}|} h^2_{j_{l_1}} \cdots h^2_{j_{l_k}} .$$

Now let $\{l_{k+1},\dots,l_d\} = \{1,\dots,d\} \backslash \{l_1,\dots,l_k\}$, then

$$\begin{aligned} \sum_{\underline{0} \leq \underline{j} \leq \underline{1}} (-4)^{|\underline{j}|} h^2_{j_{l_1}} \cdots h^2_{j_{l_k}} &= \sum_{j_1=0}^{1} \cdots \sum_{j_d=0}^{1} (-4)^{j_1+\cdots+j_d} 2^{-2(j_{l_1}+\cdots+j_{l_k})} \\ &= \prod_{m=1}^{k} \left(\sum_{j_{l_m}=0}^{1} (-1)^{j_{l_m}} \right) \prod_{m=k+1}^{d} \left(\sum_{j_{l_m}=0}^{1} (-4)^{j_{l_m}} \right) \\ &= (0)^k (-3)^{d-k} = 0 . \end{aligned}$$

Therefore all of the $C_{l_1,\dots,l_k}$ terms cancel and we are left with contributions from the $R_{\underline{i}+\underline{j}}$ terms, that is

$$\begin{aligned} |u - \tilde{u}_{\underline{i}}| &= \left| \sum_{\underline{j} \leq \underline{1}} \frac{(-4)^{|\underline{j}|}}{(-3)^d} \sum_{k=1}^{d} \sum_{\substack{\{l_1,\dots,l_k\} \\ \subset \{1,\dots,d\}}} D_{l_1,\dots,l_k}(h_{i_{l_1}+j_{l_1}},\dots,h_{i_{l_k}+j_{l_k}}) h^q_{i_{l_1}+j_{l_1}} \cdots h^q_{i_{l_k}+j_{l_k}} \right| \\ &\leq \sum_{k=1}^{d} \sum_{\substack{\{l_1,\dots,l_k\} \\ \subset \{1,\dots,d\}}} \frac{K}{3^d} h^q_{i_{l_1}} \cdots h^q_{i_{l_k}} \sum_{\underline{j} \leq \underline{1}} 4^{|\underline{j}|} h^q_{j_{l_1}} \cdots h^q_{j_{l_k}} . \end{aligned}$$

For the inner most sum letting $\{l_{k+1},\dots,l_d\} = \{1,\dots,d\} \backslash \{l_1,\dots,l_k\}$ we have

$$\begin{aligned} \sum_{\underline{0} \leq \underline{j} \leq \underline{1}} 4^{|\underline{j}|} h^q_{j_{l_1}} \cdots h^q_{j_{l_k}} &= \sum_{j_1=0}^{1} \cdots \sum_{j_d=0}^{1} 4^{j_1+\cdots+j_d} 2^{-q(j_{l_1}+\cdots+j_{l_k})} \\ &= \prod_{m=1}^{k} \left(\sum_{j_{l_m}=0}^{1} 2^{(2-q)j_{l_m}} \right) \prod_{m=k+1}^{d} \left(\sum_{j_{l_m}=0}^{1} 4^{j_{l_m}} \right) \\ &= (1+2^{2-q})^k (5)^{d-k} = 5^d \left(\frac{1+2^{2-q}}{5} \right)^k , \end{aligned}$$

which gives the desired result. □

Note that we can derive a similar extrapolation formula for $p \neq 2$, namely

$$\sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} \frac{(-2^p)^{|\underline{j}-\underline{i}|}}{(1-2^p)^d} u_{\underline{j}} .$$

It is straightforward to check that this satisfies results analogous to those in the case $p = 2$. We extend this even further by considering problems where the order of convergence may be different in each dimension. Let $p_1, \dots, p_d$ be the rate of convergence in each dimension, then the corresponding general extended error splitting model is

$$u - u_{\underline{i}} = \sum_{k=1}^{d} \sum_{\{l_1,\dots,l_k\} \subset \{1,\dots,d\}} C_{l_1,\dots,l_k} h_{i_{l_1}}^{p_{l_1}} \cdots h_{i_{l_k}}^{p_{l_k}} + R_{\underline{i}} \tag{4.12}$$

with the remainder term given by

$$R_{\underline{i}} = \sum_{k=1}^{d} \sum_{\{l_1,\dots,l_k\} \subset \{1,\dots,d\}} D_{l_1,\dots,l_k}(h_{i_{l_1}}, \dots, h_{i_{l_k}}) h_{i_{l_1}}^{q_{l_1}} \cdots h_{i_{l_k}}^{q_{l_k}}$$

with exponents $q_1 > p_1, \dots, q_d > p_d$ and constant $K > 0$ such that $|D_{l_1,\dots,l_k}| < K$ for all $\underline{i}$ and $\{l_1, \dots, l_k\}$. For this more general error splitting one has the extrapolation formula

$$\sum_{\underline{j} \leq \underline{1}} \left(\prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}} \right) u_{\underline{j}+\underline{i}} . \tag{4.13}$$

To show (4.13) do indeed result in an extrapolation we first need two lemmas.

Lemma 4.36. *One has*

$$\sum_{\underline{0} \leq \underline{j} \leq \underline{1}} \left(\prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}} \right) = 1 .$$

Proof. We observe that

$$\begin{aligned}
\sum_{\underline{0} \leq \underline{j} \leq \underline{1}} \prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}} &= \sum_{j_1=0}^{1} \cdots \sum_{j_d=0}^{1} \left(\frac{(-2^{p_1})^{j_1}}{1-2^{p_1}} \cdots \frac{(-2^{p_d})^{j_d}}{1-2^{p_d}} \right) \\
&= \left(\sum_{j_1=0}^{1} \frac{(-2^{p_1})^{j_1}}{1-2^{p_1}} \right) \cdots \left(\sum_{j_d=0}^{1} \frac{(-2^{p_d})^{j_d}}{1-2^{p_d}} \right) \\
&= \prod_{m=1}^{d} \left(\sum_{j_m=0}^{1} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}} \right) = \prod_{m=1}^{d} \left(\frac{1-2^{p_m}}{1-2^{p_m}} \right) = 1 .
\end{aligned}$$

□

Lemma 4.37. *Fix $k \in \{1,\dots,d\}$ and $\{l_1,\dots,l_k\} \subset \{1,\dots,d\}$, then*

$$\sum_{\underline{j}\leq\underline{1}} \left(\prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}}\right) h_{j_{l_1}}^{p_{l_1}} \cdots h_{j_{l_k}}^{p_{l_k}} = 0\,.$$

Proof. Letting $\{l_{k+1},\dots,l_d\} = \{1,\dots,d\}\backslash\{l_1,\dots,l_k\}$ we may swap the sum and product in a similar manner to Lemma 4.36 obtaining

$$\begin{aligned}\sum_{\underline{j}\leq\underline{1}} &\left(\prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}}\right) h_{j_{l_1}}^{p_{l_1}} \cdots h_{j_{l_k}}^{p_{l_k}} \\ &= \left(\prod_{m=1}^{k}\left(\sum_{j_{l_m}=0}^{1} \frac{(-2^{p_{l_m}})^{j_{l_m}} h_{j_{l_m}}^{p_{l_m}}}{1-2^{p_m}}\right)\right)\left(\prod_{m=k+1}^{d}\left(\sum_{j_{l_m}=0}^{1} \frac{(-2^{p_{l_m}})^{j_{l_m}}}{1-2^{p_m}}\right)\right).\end{aligned}$$

Now as $(-2^{p_{l_m}})^{j_{l_m}} h_{j_{l_m}}^{p_{l_m}} = (-1)^{j_{l_m}}$ then each of the terms in the first product is 0. As $k \geq 1$ the result follows. □

Thus we have the following proposition.

Proposition 4.38. *Let $u_{\underline{i}}$ satisfy the general extended error splitting model* (4.12) *then*

$$\left| u - \sum_{0\leq\underline{j}\leq\underline{1}} \left(\prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}}\right) u_{\underline{j}+\underline{i}} \right| \leq K \left(\prod_{m=1}^{d} \frac{1+2^{p_m}}{2^{p_m}-1}\right)\left(-1 + \prod_{k=1}^{d}(1+h_{i_k}^{q_k})\right).$$

Proof. Lemma 4.36 tells us that

$$u - \sum_{\underline{j}\leq\underline{1}} \left(\prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}}\right) u_{\underline{j}+\underline{i}} = \sum_{\underline{j}\leq\underline{1}} \left(\prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}}\right)(u - u_{\underline{j}+\underline{i}})\,.$$

On substituting the general extended error splitting (4.12) then Lemma 4.37 tells us that all of the $C_{l_1,\dots,l_k}$ cancel leaving

$$\begin{aligned}|u - \tilde{u}_{\underline{i}}| = \Bigg| &\sum_{\underline{j}\leq\underline{1}} \left(\prod_{m=1}^{d} \frac{(-2^{p_m})^{j_m}}{1-2^{p_m}}\right) \\ &\sum_{k=1}^{d} \sum_{\substack{\{l_1,\dots,l_k\}\\ \subset\{1,\dots,d\}}} D_{l_1,\dots,l_k}(h_{i_{l_1}+j_{l_1}},\dots,h_{i_{l_k}+j_{l_k}}) h_{i_{l_1}+j_{l_1}}^{q_{l_1}} \cdots h_{i_{l_k}+j_{l_k}}^{q_{l_k}} \Bigg| \\ \leq &\sum_{k=1}^{d} \sum_{\substack{\{l_1,\dots,l_k\}\\ \subset\{1,\dots,d\}}} K h_{i_{l_1}}^{q_{l_1}} \cdots h_{i_{l_k}}^{q_{l_k}} \sum_{\underline{j}\leq\underline{1}} \left(\prod_{m=1}^{d} \frac{(2^{p_m})^{j_m}}{2^{p_m}-1}\right) h_{j_{l_1}}^{q_{l_1}} \cdots h_{j_{l_k}}^{q_{l_k}}\,.\end{aligned}$$

For the inner most sum we have

$$\sum_{\underline{j}\leq\underline{1}}\left(\prod_{m=1}^{d}\frac{(2^{p_m})^{j_m}}{2^{p_m}-1}\right)h_{j_{l_1}}^{q_{l_1}}\cdots h_{j_{l_k}}^{q_{l}} \leq \sum_{\underline{j}\leq\underline{1}}\left(\prod_{m=1}^{d}\frac{(2^{p_m})^{j_m}}{2^{p_m}-1}\right)$$
$$= \prod_{m=1}^{d}\left(\sum_{j_m=0}^{1}\frac{2^{p_m j_m}}{2^{p_m}-1}\right) = \prod_{m=1}^{d}\frac{1+2^{p_m}}{2^{p_m}-1}$$

and finally using the fact that

$$\sum_{k=1}^{d}\sum_{\{l_1,\dots,l_k\}\subset\{1,\dots,d\}} h_{i_{l_1}}^{q_{l_1}}\cdots h_{i_{l_k}}^{q_{l_k}} = -1+\prod_{k=1}^{d}(1+h_{i_k}^{q_k})$$

we obtain the desired result. □

Throughout the remainder of this section we will focus on the case $p = 2$ but we observe that much of the work is easily extended to these more general cases much in the same manner the previous result have been extended from the $p = 2$ case. Given the extrapolation formula $\tilde{u}_{\underline{i}}$, as in (4.11), Reisinger goes on to consider replacing the classical combination formula

$$u_n^c := \sum_{k=0}^{d-1}(-1)^k\binom{d-1}{k}\sum_{|\underline{i}|=n-k}u_{\underline{i}}$$

with

$$\tilde{u}_n^c := \sum_{k=0}^{d-1}(-1)^k\binom{d-1}{k}\sum_{|\underline{i}|=n-k}\tilde{u}_{\underline{i}}\,.$$

His work went on to determine coefficients when $\tilde{u}_{\underline{i}}$ is expressed as a sum of $u_{\underline{i}}$ and also to derive an error bound in the $q = 4$ case, specifically

$$|u-\tilde{u}_n^c| \leq \frac{10K}{3\cdot(d-1)!}\left(\frac{85}{24}\right)^{d-1}(n+2d-2)^{d-1}2^{-4n}\,,$$

(although we point out that this result is based on the definition of a level n combination without boundaries, i.e. equivalent to the truncated combination $u_{n,\underline{1}}^t$). For the complete details we refer the reader to [109].

The work we are interested in here is the determination of combination coefficients. Reisinger [109] shows that when the combination $\tilde{u}_n^c$ is written in the form

$$\tilde{u}_n^c = \sum_{k=-d+1}^{d}\sum_{|\underline{i}|=n+k}\tilde{c}_{\underline{i}}u_{\underline{i}}\,,$$

then the coefficients are

$$\tilde{c}_{\underline{i}} = \sum_{k=\max\{0,|\underline{i}|-n-1\}}^{\min\{|\underline{i}-n+d-1,d-1|\}} \frac{(-4)^{|\underline{i}|-n+d-1-k}}{(-3)^d}(-1)^{d-1-k}\binom{d-1}{k}\binom{|\underline{i}|_0}{|\underline{i}|-n+d-1-k},$$

where $|\underline{i}|_0$ is the number of non-zero elements of $\underline{i}$. Rather than recount the proof of this we instead look at the coefficients using the framework of adaptive sparse grids that we developed in Section 4.2. This will naturally give rise to an adaptive version of the extrapolation combination technique and a simpler expression for the determination combination coefficients.

Consider linear projections $P_{\underline{i}} : V \to V_{\underline{i}}$ such that $u_{\underline{i}} = P_{\underline{i}}u$ satisfies the $p = 2, q$ extended error splitting (4.7) (here V_i must be an appropriate approximation space, for example degree $q-1$ piecewise polynomials with function values given on the usual grid points). Further suppose that $P_{\underline{i}}P_{\underline{j}} = P_{\underline{i}\wedge\underline{j}}$ for all $\underline{i}, \underline{j}$ in $\mathbb{N}^d$. We can express $\tilde{u}_{\underline{i}}$ in terms of these projections of u, specifically

$$\tilde{u}_{\underline{i}} = \left(\sum_{\underline{0}\leq\underline{j}\leq\underline{1}} \frac{(-4)^{|\underline{j}|}}{(-3)^d} P_{\underline{i}+\underline{j}}\right) u\,.$$

Thus we define the new linear projection operator

$$\tilde{P}_{\underline{i}} := \sum_{\underline{0}\leq\underline{j}\leq\underline{1}} \frac{(-4)^{|\underline{j}|}}{(-3)^d} P_{\underline{i}+\underline{j}}$$

such that $\tilde{u}_{\underline{i}} = \tilde{P}_{\underline{i}}u$. The idea now is that rather than substituting this into a classical combination formula we may substitute this into any combination formula obtained via the adaptive sparse grids formulation. That is, given a downset $I \in \mathcal{D}(\mathbb{N}^d)$ and the corresponding combination

$$P_I = \sum_{\underline{i}\in I} c_{\underline{i}} P_{\underline{i}}$$

with coefficients given by Proposition 4.21, then we define the modified version

$$\tilde{P}_I := \sum_{\underline{i}\in I} c_{\underline{i}} \tilde{P}_{\underline{i}}\,.$$

The fact that this would give a reasonable approximation follows from the fact that the $\tilde{P}_{\underline{i}}$ cancels the order p terms in the extended error splitting leaving the order q terms, and as these have the form of an error splitting the error estimate of Proposition 4.28 is easily adapted.

There are two questions that immediately come to mind regarding the choice of grids and the coefficients. Given a set $S \subset \mathbb{N}^d$ we define $S_{+\underline{1}} := \{\underline{i} : \underline{j} \in S \text{ for some } \underline{i}-\underline{1} \leq \underline{j} \leq \underline{i}\}$. For a downset I the set $I_{+\underline{1}}$ has the simpler description $I_{+\underline{1}} = \{\underline{i} : \exists \underline{j} \in I,\, \underline{i} \leq \underline{j} + \underline{1}\}$. Using this one may write

$$\tilde{P}_I = \sum_{\underline{i} \in I_{+\underline{1}}} \tilde{c}_{\underline{i}} P_{\underline{i}}\,.$$

Note that $\tilde{P}_I$ is actually a projection onto $V_{I_{+\underline{1}}}$ but as the combination is defined with respect to the downset I we use this index $\tilde{P}_I$. Additionally, in practice the computation of the $\tilde{u}_{\underline{i}} = \tilde{P}_{\underline{i}} u$ may involve down-sampling the $u_{\underline{i}+\underline{j}}$ for $\underline{0} < \underline{j} \leq \underline{1}$ similar to computations of classical Richardson extrapolation in which case the combined solution is now in the space V_I. The first question is whether there is a simple way to determine the $\tilde{c}_{\underline{i}}$ in this general formulation. The second question is more subtle and regards the possibility of extending these combinations to downsets that are not of the form $I_{+\underline{1}}$. Given a (non-empty) downset I then clearly the corresponding $I_{+\underline{1}}$ is also a downset. However, for any (non-empty) downset J we note that there is not necessarily a downset I for which $J = I_{+\underline{1}}$ (e.g. $J = \{\underline{0}\}$). The second question is therefore whether or not we may extend the definition of $\tilde{P}_I$ to sums over arbitrary (finite) downsets J. This will be addressed later as a consequence of a simple expression for the $\tilde{c}_{\underline{i}}$ which is the result of the following proposition.

Proposition 4.39. *Let $I \in \mathcal{D}(\mathbb{N}^d)$. Given the combination of extrapolations*

$$\tilde{P}_I = \sum_{\underline{i} \in I_{+\underline{1}}} \tilde{c}_{\underline{i}} P_{\underline{i}}\,,$$

which is equal to $\sum_{\underline{i} \in I} c_{\underline{i}} \tilde{P}_{\underline{i}}$, then the $\tilde{c}_{\underline{i}}$ are given by

$$\tilde{c}_{\underline{i}} = \sum_{-\underline{1} \leq \underline{l} \leq \underline{1}} \frac{(-1)^{|\underline{l}|}}{(-3)^d} 5^{d-|\underline{l}|_0} 4^{d-|\underline{l}+\underline{1}|_0} \chi_I(\underline{i}+\underline{l})$$

where $|\underline{l}|_0$ is defined to be the number of non-zero elements of $\underline{l}$.

Proof. We first write

$$\tilde{P}_I = \sum_{\underline{j} \in I} c_{\underline{j}} \tilde{P}_{\underline{j}} = \sum_{\underline{j} \in I} c_{\underline{j}} \sum_{\underline{0} \leq \underline{k} \leq \underline{1}} \frac{(-4)^{|\underline{k}|}}{(-3)^d} P_{\underline{j}+\underline{k}}$$

for which we know the $c_{\underline{j}}$ are given by

$$c_{\underline{j}} = \sum_{\underline{0} \leq \underline{k} \leq \underline{1}} (-1)^{|\underline{k}|} \chi_I(\underline{j}+\underline{k})\,. \tag{4.14}$$

The aim is to collect the $P_{\underline{i}}$ for each $\underline{i} \in I_{+\underline{1}}$. Now we note that a $\tilde{P}_{\underline{j}}$ contains $P_{\underline{i}}$ in its sum if $\underline{j} \leq \underline{i} \leq \underline{j} + \underline{1}$. Conversely, for a given $P_{\underline{i}}$, those $\tilde{P}_{\underline{j}}$ which contain $P_{\underline{i}}$ in its sum satisfy $\underline{i} - \underline{1} \leq \underline{j} \leq \underline{i}$ (and $\underline{j} \geq \underline{0}$). Further, the term for which $P_{\underline{j}+\underline{k}} = P_{\underline{i}}$ clearly satisfies $\underline{k} = \underline{i} - \underline{j}$. It follows that

$$\tilde{P}_I = \sum_{i \in I_{+\underline{1}}} P_{\underline{i}} \sum_{\substack{\underline{i}-\underline{1} \leq \underline{j} \leq \underline{i} \\ \underline{j} \geq \underline{0}}} c_{\underline{j}} \frac{(-4)^{|\underline{i}-\underline{j}|}}{(-3)^d}$$

and therefore

$$\tilde{c}_{\underline{i}} = (-3)^{-d} \sum_{\substack{\underline{i}-\underline{1} \leq \underline{j} \leq \underline{i} \\ \underline{j} \geq \underline{0}}} c_{\underline{j}} (-4)^{|\underline{i}-\underline{j}|} .$$

Now substituting (4.14) for $c_{\underline{j}}$ we obtain

$$\begin{aligned} \tilde{c}_{\underline{i}} &= (-3)^{-d} \sum_{\substack{\underline{i}-\underline{1} \leq \underline{j} \leq \underline{i} \\ \underline{j} \geq \underline{0}}} (-4)^{|\underline{i}-\underline{j}|} \sum_{\underline{0} \leq \underline{k} \leq \underline{1}} (-1)^{|\underline{k}|} \chi_I(\underline{j} + \underline{k}) \\ &= (-3)^{-d} \sum_{\substack{(\underline{0} \leq)\, \underline{j} \leq \underline{1} \\ \underline{j} \leq \underline{i}}} (-4)^{|\underline{j}|} \sum_{\underline{0} \leq \underline{k} \leq \underline{1}} (-1)^{|\underline{k}|} \chi_I(\underline{i} - \underline{j} + \underline{k}) , \end{aligned}$$

where the last line follows from the substitution $\underline{j} \mapsto \underline{i} - \underline{j}$. We now make another substitution $\underline{l} = \underline{k} - \underline{j}$, notice that we allow $\underline{l}$ to take on negative values,

$$\begin{aligned} \tilde{c}_{\underline{i}} &= (-3)^{-d} \sum_{\substack{(\underline{0} \leq)\, \underline{j} \leq \underline{1} \\ \underline{j} \leq \underline{i}}} (-4)^{|\underline{j}|} \sum_{-\underline{j} \leq \underline{l} \leq \underline{1} - \underline{j}} (-1)^{|\underline{l}+\underline{j}|} \chi_I(\underline{i} + \underline{l}) \\ &= (-3)^{-d} \sum_{-\underline{1} \leq \underline{l} \leq \underline{1}} \chi_I(\underline{i} + \underline{l}) \sum_{\max\{\underline{0}, -\underline{l}\} \leq \underline{j} \leq \min\{\underline{1}, \underline{1}-\underline{l}\}} (-4)^{|\underline{j}|} (-1)^{|\underline{l}+\underline{j}|} . \end{aligned}$$

The second line here is a change of order of summation and the min, max over the multi-indices are component wise. We now consider the inner summation, noting that $\underline{j} \geq \underline{0}$ and $\underline{l} + \underline{j} \geq \underline{0}$ we can write this as

$$\begin{aligned} &\sum_{\max\{\underline{0}, -\underline{l}\} \leq \underline{j} \leq \min\{\underline{1}, \underline{1}-\underline{l}\}} (-4)^{j_1 + \cdots + j_d} (-1)^{l_1 + j_1 + \cdots + l_d + j_d} \\ &\quad = (-1)^{l_1 + \cdots + l_d} \sum_{\max\{\underline{0}, -\underline{l}\} \leq \underline{j} \leq \min\{\underline{1}, \underline{1}-\underline{l}\}} 4^{j_1} \cdots 4^{j_d} \\ &\quad = (-1)^{l_1 + \cdots + l_d} \left(\sum_{j_1 = \max\{0, -l_1\}}^{\min\{1, 1-l_1\}} 4^{j_1} \right) \cdots \left(\sum_{j_d = \max\{0, -l_d\}}^{\min\{1, 1-l_d\}} 4^{j_d} \right) \\ &\quad = (-1)^{l_1 + \cdots + l_d} 5^{d - |\underline{l}|_0} 4^{d - |\underline{l}+\underline{1}|_0} , \end{aligned}$$

where $d - |\underline{l}|_0$ and $d - |\underline{l} + \underline{1}|_0$ are the number of elements of $\underline{l}$ which are 0 and -1 respectively. (Note that when an element of $\underline{l}$ is 1 the corresponding sum is 1). Finally we note that $(-1)^{l_1+\cdots+l_d} = (-1)^{|\underline{l}|}$ since for any $l_k = -1$ we have $(-1)^{l_k} = (-1)^{l_k+2} = (-1)^{-l_k}$. Substituting this back into equation for $\tilde{c}_{\underline{i}}$ yields the desired result. □

As a result of this proposition we can compute coefficients for an adaptive extrapolation approach very quickly. The following proposition leads to a consistency property for the $\tilde{P}_I$.

Proposition 4.40. *Let $I \in \mathcal{D}(\mathbb{N}^d)$ (non-empty) and $\tilde{P}_I$ be the combination of extrapolations*

$$\tilde{P}_I = \sum_{\underline{i} \in I_{+\underline{1}}} \tilde{c}_{\underline{i}} P_{\underline{i}} \,,$$

with coefficients $\tilde{c}_{\underline{i}}$ given by Proposition 4.39, then for $\underline{i} \in I$ one has $P_{\underline{i}} \tilde{P}_I = P_{\underline{i}}$, or equivalently

$$\sum_{\underline{j} \geq \underline{i}} \tilde{c}_{\underline{j}} = 1 \,.$$

Proof. We note that

$$\begin{aligned} P_{\underline{i}} \tilde{P}_I &= \sum_{\underline{j} \in I_{+\underline{1}}} \tilde{c}_{\underline{j}} P_{\underline{i}} P_{\underline{j}} \\ &= \sum_{\substack{\underline{j} \in I_{+\underline{1}} \\ \underline{j} \geq \underline{i}}} \tilde{c}_{\underline{j}} P_{\underline{i}} \,. \end{aligned}$$

Now writing the $\tilde{c}_{\underline{j}}$ in terms of $c_{\underline{j}}$ we have

$$\sum_{\underline{j} \geq \underline{i}} \tilde{c}_{\underline{j}} = \sum_{\underline{j} \geq \underline{i}} c_{\underline{j}} \sum_{\underline{0} \leq \underline{k} \leq \underline{1}} \frac{(-4)^{|\underline{k}|}}{(-3)^d} \,,$$

and as the inner sum is 1 (Lemma 4.34) we have

$$\sum_{\underline{j} \geq \underline{i}} \tilde{c}_{\underline{j}} = \sum_{\underline{j} \geq \underline{i}} c_{\underline{j}} = \chi_I(\underline{i}) = 1$$

as a consequence of Lemma 4.20. □

Corollary 4.41. *Let $I \in \mathcal{D}(\mathbb{N}^d)$ (non-empty) and $\tilde{P}_I$ be the combination of extrapolations*

$$\tilde{P}_I = \sum_{\underline{i} \in I_{+\underline{1}}} \tilde{c}_{\underline{i}} P_{\underline{i}} \,,$$

then

$$\sum_{\underline{i}\in I_{+\underline{1}}} \tilde{c}_{\underline{i}} = 1\,.$$

Proof. Simply apply Proposition 4.40 to the case $\underline{i} = \underline{0}$. □

Now that we have addressed the question regarding the coefficients we turn to the second question regarding the downsets over which we combine. When a covering element $\underline{i}$ with one or more zero components is added to a downset I to form $J = I \cup \{\underline{i}\}$ then $J_{+\underline{1}}$ is not simply $I_{+\underline{1}} \cup \{\underline{i}+\underline{1}\}$ (as is the case if $\underline{i} > \underline{0}$). For example, for $d = 2$ given $I = \{(0,0)\}$, and hence $I_{+\underline{1}} = \{(1,1)\}{\downarrow}$), then for $J = I \cup \{(1,0)\}$ we have $J_{+\underline{1}} = I_{+\underline{1}} \cup \{(2,1),(2,0)\}$ and thus two additional projections are required to compute $\tilde{P}_J$ compared to $\tilde{P}_I$. More generally the number of additional projections required is $2^{d-|\underline{i}|_0}$ where $\underline{i}$ is the covering element added to I. The question that arises is whether we can modify the combination to act on any downset so that only 1 element needs to be added to the combination at each stage. Let us rewrite the $\tilde{c}_{\underline{i}}$ of $\tilde{P}_I$ by shifting $\underline{l}$ by $\underline{1}$ as follows

$$\begin{aligned}
\tilde{c}_{\underline{i}} &= \sum_{-\underline{1}\le \underline{l}-\underline{1}\le \underline{1}} \frac{(-1)^{|\underline{l}-\underline{1}|}}{(-3)^d} 5^{d-|\underline{l}-\underline{1}|_0} 4^{d-|\underline{l}|_0} \chi_I(\underline{i}+\underline{l}-\underline{1}) \\
&= \sum_{\underline{0}\le \underline{l}\le \underline{2}} \frac{(-1)^{|\underline{l}|}}{3^d} 5^{d-|\underline{l}-\underline{1}|_0} 4^{d-|\underline{l}|_0} \chi_{I_{+\underline{1}}}(\underline{i}+\underline{l})\,.
\end{aligned}$$

Now given a (non-empty, finite) downset J we consider the combination

$$\tilde{\tilde{P}}_J := \sum_{\underline{i}\in J} \tilde{\tilde{c}}_{\underline{i}} P_{\underline{i}} u$$

where the coefficients are given by

$$\tilde{\tilde{c}}_{\underline{i}} := \sum_{\underline{0}\le \underline{l}\le \underline{2}} \frac{(-1)^{|\underline{l}|}}{3^d} 5^{d-|\underline{l}-\underline{1}|_0} 4^{d-|\underline{l}|_0} \chi_J(\underline{i}+\underline{l})\,.$$

If there exists a downset I such that $J = I_{+\underline{1}}$ then $\tilde{\tilde{P}}_J = \tilde{P}_I$. Our second question can now be rephrased as whether $\tilde{\tilde{P}}_J$ is a reasonable projection if there does not exist a downset I such that $J = I_{+\underline{1}}$. By reasonable we specifically mean two things, first the coefficients should sum to 1 to provide consistency, and second, the order $p = 2$ error terms should sum to zero when the $P_{\underline{i}}u$ satisfies the extended error splitting model. Observations and experiments seem to indicate that the resulting combination is reasonable if $\underline{1} \in J$ (and thus $\{\underline{1}\}{\downarrow} \subset J$). This is not too surprising as the multi-variate extrapolation applied to a single $u_{\underline{i}}$ requires those $u_{\underline{j}}$ with $\underline{i} \leq \underline{j} \leq \underline{i} + \underline{1}$. We suspect this is both a sufficient and necessary condition. This is relatively straightforward to show in 2 dimensions by breaking the problem up into the few cases that can occur. It remains an open problem in higher dimensions.

This lends itself to a more general adaptive scheme, and analogous to the regular adaptive sparse grids we could ask what $\tilde{\tilde{P}}_J - \tilde{\tilde{P}}_I$ looks like for a cover J of I.

Corollary 4.42. *Let $I \in \mathcal{D}(\mathbb{N}^d)$ with $\underline{1} \in I$ and let $\underline{i}$ be a covering element of I. Let $J = I \cup \{\underline{i}\}$, then*

$$\tilde{\tilde{P}}_J - \tilde{\tilde{P}}_I = \sum_{\substack{\underline{i}-\underline{2} \leq \underline{j} \leq \underline{i} \\ \underline{j} \geq \underline{0}}} \frac{(-1)^{|\underline{i}-\underline{j}|}}{3^d} 5^{d-|\underline{i}-\underline{j}-\underline{1}|_0} 4^{d-|\underline{i}-\underline{j}|_0} P_{\underline{j}}$$

Proof. This is an immediate consequence of Proposition 4.39. □

For the general extrapolation formula (4.13) one may follow the same procedure to obtain the formulas

$$\tilde{c}_{\underline{i}} = \sum_{-\underline{1} \leq \underline{l} \leq \underline{1}} (-1)^{|\underline{l}|} \chi_I(\underline{i} + \underline{l}) \left(\prod_{m=1}^{d} \frac{\delta_{-1,l_m} 2^{p_m} + \delta_{0,l_m}(1 + 2^{p_m}) + \delta_{1,l_m}}{1 - 2^{p_m}} \right),$$

(with $\delta_{a,b} = 1$ if $a = b$ and $\delta_{a,b} = 0$ otherwise), and

$$\tilde{\tilde{P}}_{\underline{J}} - \tilde{\tilde{P}}_{\underline{I}} = \sum_{\substack{\underline{i}-\underline{2} \leq \underline{j} \leq \underline{i} \\ \underline{j} \geq \underline{0}}} (-1)^{|\underline{i}-\underline{j}|} \left(\prod_{m=1}^{d} \frac{\delta_{i_m-2,j_m} + \delta_{i_m-1,j_m}(1 + 2^{p_m}) + \delta_{i_m,j_m} 2^{p_m}}{2^{p_m} - 1} \right) P_{\underline{j}}$$

for Proposition 4.39 and Corollary 4.42 respectively.

4.4 The General Coefficient Problem

In Section 4.2 we considered combinations over sets of indices $I \subset \mathbb{N}^d$ which were downsets. In such cases we had the nice framework of projections onto function space lattices which allowed us to derive simple expressions for the combination coefficients. In this section we are concerned with the determination of combination coefficients given any finite $I \subset \mathbb{N}^d$ such that

$$\sum_{\underline{i} \in I} c_{\underline{i}} u_{\underline{i}}$$

is a good approximation to u. The motivation for this is that on a computer in which faults occur, one may intend to compute $u_{\underline{i}}$ for $\underline{i}$ in some downset I, but if the computation of some of these are affected by faults then the subset of I for which the $u_{\underline{i}}$ were computed successfully may no longer be a downset. In such circumstances it would not be clear how these remaining $u_{\underline{i}}$ could be combined. Thus our aim is to develop new ways to compute coefficients for such circumstances. Further, we would like the coefficients to be the best possible in some sense.

Opticom [71] is a generalisation of the combination technique that is applicable to minimisation problems. For example, given a convex function $\mathcal{J} : V \to \mathbb{R}$ and a numerical scheme which finds the $v \in V_{\underline{i}} \subset V$ which minimises $\mathcal{J}$, that is $\operatorname{argmin}_{v \in V_{\underline{i}}} \mathcal{J}(v)$, then given approximations $u_{\underline{i}}$ for $\underline{i} \in I$ opticom is the problem of finding the $c_{\underline{i}}$ such that $\mathcal{J}\left(\sum_{\underline{i} \in I} c_{\underline{i}} u_{\underline{i}}\right)$ is minimised. It is clear that this leads to the best possible outcome given the $u_{\underline{i}}$ available. Opticom is an example of a naturally fault tolerant algorithm as any faults in the $u_{\underline{i}}$ are automatically handled through the minimisation process. For hyperbolic PDEs it is not typical to have such a function to minimise. An exception is when solutions are obtained via the least squares finite element method [12] but we will not consider this here.

We consider the problem of minimising $\left\| u - \sum_{\underline{i} \in I} c_{\underline{i}} u_{\underline{i}} \right\|$ with respect to an appropriate norm. As the exact function u is generally not available in practice and finite difference methods generally do not maintain an estimate of the residual it is not feasible to minimise this directly. However, in Section 4.2 we obtained bounds on the 2 and ∞ norms of this quantity for interpolation of $u \in H^2_{0,\text{mix}}$ and for $u_{\underline{i}}$ satisfying the error splitting model respectively. Thus we consider two different approaches for computing combination coefficients based on these a priori error bounds. By choosing coefficients which minimise these bounds we can be reasonably confident that the result will be close to the best possible, provided the bounds are tight. Of course, the bounds obtained in Section 4.2 applied to $I \subset \mathbb{N}^d$ which were downsets and we therefore need to find a way to extend these results to arbitrary I.

4.4.1 Combination Coefficients via Interpolation Bounds

In Proposition 4.26 we showed that for $u \in H^2_{0,\text{mix}}([0,1]^d)$ and $u_{\underline{i}}$ which are piecewise linear interpolants of u between the function values on the grid points $\Omega_{\underline{i}} = \{(j_1 2^{-i_1}, \dots, j_d 2^{-i_d}) : \underline{0} \leq \underline{j} \leq 2^{\underline{i}}\}$, downsets $I \subset \mathbb{N}^d_+$, and corresponding combination coefficients given by

$$c_{\underline{i}} = \sum_{\underline{i} \leq \underline{j} \leq \underline{i}+\underline{1}} (-1)^{|\underline{j}-\underline{i}|} \chi_I(\underline{j}) \,,$$

see Proposition 4.21, then one has

$$\left\| u - \sum_{\underline{i} \in I} c_{\underline{i}} u_{\underline{i}} \right\|_2 \leq 3^{-d} \left\| D^{\underline{2}} u \right\|_2 \left(3^{-d} - \sum_{\underline{1} \leq \underline{i} \in I} 2^{-2|\underline{i}|} \right) . \tag{4.15}$$

Given an arbitrary (finite, non-empty) $J \subset \mathbb{N}^d$ if we can find a downset $I \subset J{\downarrow}$ such that the coefficients corresponding to the combination over I are zero on $J \backslash (I \cap J)$ then the same combination can be computed over the set J. Further, the error of this combination over J will satisfy the bound (4.15) for the corresponding I. We note that there is always at least one such I, for example take $I = \{\underline{i}\}{\downarrow}$ for any $\underline{i} \in J$. Thus in order to minimise the bound we need to choose a suitable I which maximises $\sum_{\underline{1} \leq \underline{i} \in I} 2^{-2|\underline{i}|}$. We note that there is typically not a unique I which satisfies these criteria.

Example 4.43. Let $d = 2$ and $J = \{(2,1),(1,2)\}$. There are only two possible (non-empty) combinations one may obtain which correspond to $P_{(1,2)\downarrow}$ and $P_{(2,1)\downarrow}$. It is clear that $2^{-2|(1,1)|} + 2^{-2|(2,1)|} = \frac{5}{64} = 2^{-2|(1,1)|} + 2^{-2|(1,2)|}$. It is not possible to determine if one of these is better than the other without additional information about u.

Another point is that we typically deal with functions which are not zero on the boundaries. We could extend the result of Proposition 2.22 to the adaptive sparse grids setting to specifically handle this scenario but we will instead take the approach of simply extending the sum to $\underline{i} \in I$ (i.e. removing the restriction $\underline{i} \geq \underline{1}$) so that we choose I which maximises $\sum_{\underline{i} \in I} 2^{-2|\underline{i}|}$.

To summarise, we define our problem as follows.

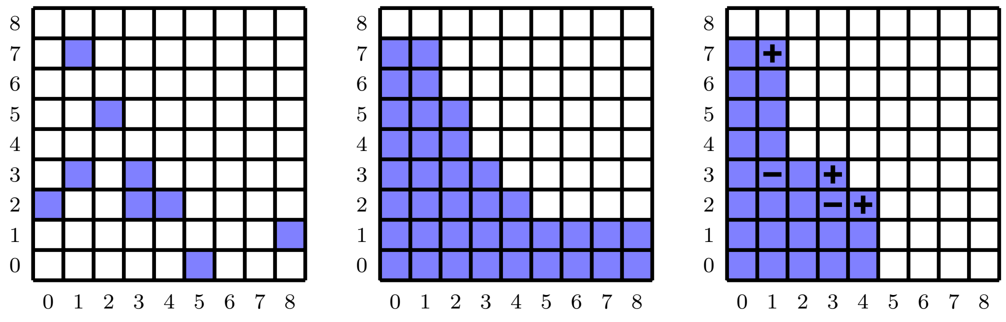


Figure 4.5: *Suppose $J \subset \mathbb{N}^2$ is the set of elements marked in blue in the diagram on the left, then for the general coefficient problem we need to find a downset $I \leq J{\downarrow}$ such that the combination coefficients generated by I are only non-zero on the set J. The set $J{\downarrow}$ is depicted in the centre and on the right is a candidate I with the corresponding combination coefficients. This particular I solves the* GCP *for the set J.*

Definition 4.44. Let $J \subset \mathbb{N}^d$ be finite and non-empty. The *general coefficient problem* (GCP) for the set J is the problem of finding a downset $I \leq J{\downarrow}$ with coefficients $\{c_{\underline{i}}\}_{\underline{i}\in I}$ corresponding to the projection

$$P_I = \sum_{\underline{i}\in I} c_{\underline{i}} P_{\underline{i}}$$

which satisfy $c_{\underline{i}} = 0$ for $\underline{i} \in I \backslash (J \cap I)$ and I maximises

$$\mathcal{J}(I) := \sum_{\underline{i}\in I} 2^{-2|\underline{i}|} . \tag{4.16}$$

We denote GCP(J) to be the set of solutions for a given J. An example is given in Figure 4.5.

Where $|\text{GCP}(J)| > 1$ we cannot say any one solution in this set is better than another unless we have more information about u. There are a few approaches we can take in such cases. The first option is to randomly pick a $I \in \text{GCP}(J)$ and be satisfied that any such one minimises the bound on interpolation error. A second option is to compute the combinations $u_I = \sum_{\underline{i}\in I} c_{\underline{i}} u_{\underline{i}}$ for each $I \in \text{GCP}(J)$ and then analyse the hierarchical surpluses of each to determine which contribute the most. A third option is to again compute u_I for each $I \in \text{GCP}(J)$ but this time we pick one such solution, say $u_{I'}$ and add to it the hierarchical surpluses from the remaining $u_{\underline{i}}$ which do not contribute to $u_{I'}$. In practice we typically choose the first option as it is cheaper than the other two options (requiring only one combination to be computed).

Whilst Definition 4.44 gives a nice description of the problem we are trying to solve it does not give a clear indication of how solutions may be computed. We address this by reformulating the problem. We start by defining the hierarchical coefficient.

Definition 4.45. Let $c_{\underline{i}} \in \mathbb{R}$ for each $\underline{i} \in \mathbb{N}^d$, then we define the hierarchical coefficients to be

$$\omega_{\underline{i}} := \sum_{\underline{j} \geq \underline{i}} c_{\underline{j}} .$$

The motivation for this definition is that given $u_{\underline{i}}$ which are the piecewise multi-linear interpolants of some function u in the usual function space lattice (e.g. (2.2)), then given the hierarchical surpluses $u_{\underline{i}}^h$, a finite $I \subset \mathbb{N}^d$ and combination coefficients $c_{\underline{i}}$ for $\underline{i} \in I$, then with $c_{\underline{i}} := 0$ for $\underline{i} \notin I$ one has

$$\sum_{\underline{i} \in I} c_{\underline{i}} u_{\underline{i}} = \sum_{\underline{i} \in \mathbb{N}^d} \omega_{\underline{i}} u_{\underline{i}}^h .$$

Thus each hierarchical contribution $\omega_{\underline{i}}$ tells us how many times the corresponding hierarchical surplus $u_{\underline{i}}^h$ contributes to the approximation. If I is a finite downset and the c_i are the coefficients corresponding to the projection P_I then as a consequence of Lemma 4.20 we have $\omega_{\underline{i}} = \chi_I(\underline{i})$. Thus, in this case, we can also write the $c_{\underline{i}}$ in terms of the $\omega_{\underline{i}}$ via

$$c_{\underline{i}} = \sum_{\underline{i} \leq \underline{j} \leq \underline{i} + \underline{1}} (-1)^{|\underline{j} - \underline{i}|} \omega_{\underline{j}} .$$

Now let $\boldsymbol{c}, \boldsymbol{\omega}$ be vectors for $\{c_{\underline{i}}\}_{\underline{i} \in I}, \{\omega_{\underline{i}}\}_{\underline{i} \in I}$ respectively (with the same ordering). It follows we can define a matrix M such that $\boldsymbol{\omega} = M\boldsymbol{c}$, in particular, it follows from the definition that given indices m, n and $\underline{i}, \underline{j}$ such that $\boldsymbol{c}_m = c_{\underline{i}}$ and $\boldsymbol{c}_n = c_{\underline{j}}$ (and similarly for $\boldsymbol{\omega}$) then $M_{m,n} = 1$ if $\underline{i} \leq \underline{j}$ and $M_{m,n} = 0$ otherwise. If the entries in the vectors $\boldsymbol{c}, \boldsymbol{\omega}$ are ordered according to non-decreasing values of $|\underline{i}|$ then M is clearly upper triangular with 1's on the main diagonal. It follows that M is invertible and we can write $\boldsymbol{c} = M^{-1}\boldsymbol{\omega}$. Further, the structure of M^{-1} is given by the result of Proposition 4.21, namely given positive integers m, n and multi-indices $\underline{i}, \underline{j}$ such that $\boldsymbol{c}_m = c_{\underline{i}}$ and $\boldsymbol{c}_n = c_{\underline{j}}$ (and similarly for $\boldsymbol{\omega}$) then $(M^{-1})_{m,n} = (-1)^{|\underline{j} - \underline{i}|}$ if $\underline{i} \leq \underline{j} \leq \underline{i} + \underline{1}$ and $(M^{-1})_{m,n} = 0$ otherwise. It is clear that this too is upper triangular and all entries are elements of the set $\{-1, 0, 1\}$.

There are two important observations I will emphasise at this point. First, the hierarchical coefficients, $\omega_{\underline{i}}$, are always 0 or 1 for combinations which correspond to adaptive sparse grids. Therefore we can consider the hierarchical coefficients to be binary variables if we are only interested in such combinations. Second, as M^{-1} has integer coefficients then it is a direct consequence that the combination coefficients, $c_{\underline{i}}$, are integers.

Given an arbitrary (finite) $J \subset \mathbb{N}^d$ we form the vectors $\boldsymbol{c} = \{c_{\underline{i}}\}_{J\downarrow}$ and $\boldsymbol{\omega} = \{\omega_{\underline{i}}\}_{J\downarrow}$, and the matrix M with entries as described above. Our goal is to find a downset $I \leq J\downarrow$ such that I maximises $\mathcal{J}(I)$ and the resulting coefficients are zero on $I \backslash (J \cap I)$. For $I \leq J\downarrow$ we observe that the resulting hierarchical coefficients satisfy $\omega_{\underline{i}} = 1$ for $\underline{i} \in I$ and $\omega_{\underline{i}} = 0$ for $\underline{i} \in J\downarrow \backslash I$. Further, the condition that the resulting $c_{\underline{i}}$ are zero on $I \backslash (J \cap I)$ and hence are zero on $J\downarrow \backslash J$ is equivalent to the condition $(M^{-1}\boldsymbol{\omega})_m = 0$ for each m such that $\boldsymbol{\omega}_m$ corresponds to an $\omega_{\underline{i}}$ for which $\underline{i} \in I\downarrow \backslash I$. To simplify notation we denote $m_{\underline{i}}$ as the index corresponding to $\boldsymbol{\omega}_{m_{\underline{i}}} = \omega_{\underline{i}}$ (and $\boldsymbol{c}_{m_{\underline{i}}} = c_{\underline{i}}$). We can rewrite $\mathcal{J}(I)$ for downsets $I \leq J\downarrow$ in terms of the $\omega_{\underline{i}}$ as

$$\sum_{\underline{i} \in J\downarrow} 4^{-|\underline{i}|} \omega_{\underline{i}} \,.$$

Thus, putting all of this together we have the following lemma.

Lemma 4.46. *Given finite $J \subset \mathbb{N}^d$ then $I \in$ GCP(J) if and only if $\{\omega_{\underline{i}} = \chi_I(\underline{i})\}_{\underline{i} \in J\downarrow}$ is a solution to the binary integer programming (BIP) problem of finding $\boldsymbol{\omega} = \{\omega_{\underline{i}}\}_{\underline{i} \in J\downarrow} \in \{0,1\}^{|J\downarrow|}$ which maximises*

$$\mathcal{J}_2(\boldsymbol{\omega}) = \sum_{\underline{i} \in J\downarrow} 4^{-|\underline{i}|} \omega_{\underline{i}} \tag{4.17}$$

subject to the equality constraints $(M^{-1}\boldsymbol{\omega})_{m_{\underline{i}}} = 0$ for $\underline{i} \in J\downarrow \backslash J$ and the inequality constraints $\omega_{\underline{j}} \geq \omega_{\underline{i}}$ for all $\underline{j} \leq \underline{i}$.

Proof. By construction we have a one to one correspondence with I which solve the GCP and $\boldsymbol{\omega}$ which solves the BIP described. □

Remark 4.47. We depart for a moment to remark on the inequality restraints $\omega_{\underline{i}} \geq \omega_{\underline{j}}$ for all $\underline{i} \leq \underline{j}$. The inclusion of this restraint is necessary to ensure that the set

$$\{\underline{i} \in J\downarrow : \omega_{\underline{i}} = 1\}$$

is a downset which is a requirement for the solutions of the GCP. Typically the solution to the BIP formulation without this constraint naturally leads to a downset. However, there are some J for which this is not so. For example, consider in two dimensions the set $I = \{(3,0),(0,1),(0,2)\}$. It is clear that the solution to the GCP is given by $(3,0)\downarrow$. On the other hand, the solution to the

BIP formulation without the downset constraint is given by $\omega_{\underline{i}}$ which are 1 for $\underline{i} \in (3,0){\downarrow} \cup (0,2)$. It is straightforward to check that this satisfies the equality constraints and maximises the sum. However, clearly this is not a downset and therefore one would typically dismiss it as being a sensible combination. That said, in some circumstances this may yield a reasonable result, particularly in cases where the $u_{\underline{i}}$ are exactly the piecewise multi-linear interpolants of u.

The above BIP formulation can be solved by listing all downsets $I \leq J{\downarrow}$, filtering out those which do not satisfy the equality constraints, and then evaluating which of the remaining downsets achieves the maximum. This exhaustive approach is typically not feasible due to the number of possible downsets that one may be required to check which grows incredibly fast with respect to the size of J. The optimisation problem can also be solved using branch and bound and/or cutting plane techniques. None the less, the formulation as a constrained BIP problem reveals that the GCP is NP-complete [77]. As a result, given any algorithm there are always sets J which will take a long time to be solved compared to other sets of similar size. This is also why in practice we typically choose a random $I \in \text{GCP}(J)$ to use as the solution as finding the exhaustive set of solutions $\text{GCP}(J)$ can be significantly more expensive than finding just 1. It is worth noting there are some J for which the solution is trivial to compute. For example if J is a downset (i.e. $J = J{\downarrow}$) then we can solve this rather quickly with the coefficients simply being those given by the projection $P_J = P_{J\downarrow}$. Similarly if J is closed under meet then we know that the projection $P_{J\downarrow}$ produces coefficients which are non-zero only for multi-indices $\underline{i} \in J$ and thus solutions are again quick to compute in such cases.

Whilst the GCP is based upon an error estimate for the interpolation of $u \in H^2_{\text{mix}}$ it is applicable to a much more broad range of problems. First we observe that if the $u_{\underline{i}}$ satisfy the error splitting model (2.22) then the error bound of Proposition 4.28 for adaptive sparse grids is applicable to the solutions of the GCP. Further, as solutions to the GCP are adaptive sparse grids then they should perform well in the wide range of applications for which adaptive sparse grids have been studied.

Combinations with real coefficients

We can also consider an alternative approach for which the hierarchical coefficients are reals (and thus the combination coefficients are also reals). This first requires us to extend the result of Proposition 4.26 to arbitrary combination formula.

Proposition 4.48. *Let $u \in H^2_{0,mix}([0,1]^d)$ and let $u_{\underline{i}}$ be piecewise linear interpolants of u as defined in (2.2). Given $J \subset \mathbb{N}^d$ which is finite and non-empty, and real coefficients $\{c_{\underline{i}}\}_{\underline{i} \in J}$ then*

$$\left\| u - \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}} \right\|_2 \leq 3^{-d} \left\| D^{\underline{2}} u \right\|_2 \sum_{\underline{j} \geq \underline{1}} 2^{-2|\underline{j}|} \left| 1 - \sum_{\underline{i} \in J \text{ s.t. } \underline{i} \geq \underline{j}} c_{\underline{j}} \right| . \tag{4.18}$$

Proof. The $u_{\underline{i}}$ may be expressed as $u_{\underline{i}} = \sum_{\underline{j} \leq \underline{i}} u^h_{\underline{j}}$ where $u^h_{\underline{j}}$ is the level $\underline{j}$ hierarchical surplus of u (see Section 2.1). Thus as u is zero on the boundaries one has

$$\begin{aligned} u - \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}} &= \left(\sum_{\underline{j} \geq \underline{1}} u^h_{\underline{j}} \right) - \sum_{\underline{i} \in J} c_{\underline{i}} \left(\sum_{\underline{1} \leq \underline{j} \leq \underline{i}} u^h_{\underline{j}} \right) \\ &= \sum_{\underline{j} \geq \underline{1}} u^h_{\underline{j}} \left(1 - \sum_{\underline{i} \in J \text{ s.t. } \underline{i} \geq \underline{j}} c_{\underline{i}} \right) . \end{aligned}$$

Using the triangle inequality it follows that

$$\left\| u - \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}} \right\|_2 \leq \sum_{\underline{j} \geq \underline{1}} \left\| u^h_{\underline{j}} \right\|_2 \left| 1 - \sum_{\underline{i} \in J \text{ s.t. } \underline{i} \geq \underline{j}} c_{\underline{i}} \right| .$$

Inserting the bound $\|u^h_{\underline{j}}\|_2 \leq 3^{-d} \left\| D^{\underline{2}} u \right\|_2 2^{-2|\underline{j}|}$ from Lemma 2.14 and moving the common terms to the left of the sum gives the desired result. □

Here, if we let $c_{\underline{i}} = 0$ for $\underline{i} \notin J$ then we can rewrite this result in terms of our hierarchical coefficients $\omega_{\underline{i}} = \sum_{\underline{j} \geq \underline{i}} c_{\underline{j}}$ as

$$\left\| u - \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}} \right\|_2 \leq 3^{-d} \left\| D^{\underline{2}} u \right\|_2 \sum_{\underline{j} \geq \underline{1}} 2^{-2|\underline{j}|} \left| 1 - \omega_{\underline{j}} \right| .$$

The idea is to now choose coefficients which minimise this bound on the error. We again drop the restriction $\underline{j} \geq \underline{1}$ to extend the problem beyond functions which are zero on the boundary. Further, as the only coefficients we can control are those in J then we are really only concerned with minimising over $\underline{j} \in J\downarrow$. As before we form the vectors $\boldsymbol{c} = \{c_{\underline{i}}\}_{J\downarrow}$, $\boldsymbol{\omega} = \{\omega_{\underline{i}}\}_{J\downarrow}$ and the matrix M for which

$\boldsymbol{\omega} = M\boldsymbol{c}$. The alternative approach now consists of solving the L_1 minimisation problem of finding $\boldsymbol{\omega} \in \mathbb{R}^{|J\downarrow|}$ which minimises

$$\mathcal{J}_3(\boldsymbol{\omega}) := \sum_{\underline{j}\in J\downarrow} 2^{-2|\underline{j}|} \left|1 - \omega_{\underline{j}}\right| \tag{4.19}$$

subject to the equality constraints $(M^{-1}\boldsymbol{\omega})_m = 0$ for any index m corresponding to $\underline{i} \in J\downarrow \backslash J$.

It is clear that the solutions to this problem lead to an error bound which is no larger than that of the GCP solutions. Quite often this alternative approach produces solutions identical to the GCP. However this is not always the case as we show in the following example.

Example 4.49. Consider the two dimensional problem with $J = \{(1,0),(0,1)\}$. Clearly $\text{GCP}(J) = \{\{(1,0)\}\downarrow, \{(0,1)\}\downarrow\}$ leading to the combinations $u_{(1,0)}$ and $u_{(0,1)}$. On the other hand, it is easily shown that the solutions to the alternative problem correspond to the combinations $\alpha u_{(1,0)} + (1-\alpha)u_{(0,1)}$ for $\alpha \in [0,1]$.

In our experience, whilst the combinations produced by this alternative formulation work well for interpolation they are not as robust for other applications (e.g. problems satisfying an error splitting) when the solutions differ from those of the GCP. Nonetheless, this L_1 minimisation problem can be used as a first attempt at finding solutions to the GCP. If the $\boldsymbol{\omega}$ which solves the L_1 minimisation problem has entries which are all $\{0,1\}$ then we can take this to be a solution to the GCP provided the downset condition is also satisfied. Otherwise, we can find the binary vector closest to the given solution and use this as an initial guess in the GCP solver.

4.4.2 Combination Coefficients via Error Splitting Bounds

Here we consider finding coefficients which minimise a bound on the combination error when the $u_{\underline{i}}$ are assumed to satisfy an error splitting. So far we have considered two different error splitting models, that is (2.22) and the extended version (4.7), and the generalisations (4.6) and (4.12) respectively. We consider both here as they typically lead to different combinations. Proposition 4.28 bound the error for adaptive sparse grid combinations over $u_{\underline{i}}$ which satisfied the error splitting model (2.22). Here we extend this result to arbitrary combinations so that the bound may be optimised with respect to the combination coefficients. First we remind the reader of the following notation as introduced in Section 4.2.

Given $I \subset \mathbb{N}^d$ we let

$$I_{e_1,\dots,e_k} = \{\underline{i} \in \mathbb{N}^k : \underline{i} = (j_{e_1}, \dots, j_{e_k}) \text{ for some } \underline{j} \in I\}\,,$$

and for $\underline{l} \in \mathbb{N}^k$ we let

$$I_{\underline{l}|e_1,\dots,e_k} = \{\underline{i} \in I : (i_{e_1}, \dots, i_{e_k}) = (l_1, \dots, l_k)\}\,.$$

such that

$$\sum_{\underline{i} \in I} f(\underline{i}) = \sum_{\underline{j} \in I_{e_1,\dots,e_k}} \left(\sum_{\underline{l} \in I_{\underline{j}|e_1,\dots,e_k}} f(\underline{i}) \right)\,. \tag{4.20}$$

Proposition 4.50. *Let $u \in C([0,1]^d)$ and $J \subset \mathbb{N}^d$ be finite and non-empty. For each $\underline{i} \in J$ let $c_{\underline{i}} \in \mathbb{R}$ such that $\sum_{\underline{i} \in J} c_{\underline{i}} = 1$ and let each $u_{\underline{i}}$ be an approximation of u satisfying the point-wise error splitting model*

$$u - u_{\underline{i}} = \sum_{k=1}^{d} \sum_{\{e_1,\dots,e_k\} \subset \{1,\dots,d\}} C_{e_1,\dots,e_k}(h_{i_{e_1}}, \dots, h_{i_{e_k}}) h_{i_{e_1}}^{p_{e_1}} \cdots h_{i_{e_k}}^{p_{e_k}}\,,$$

where $p_1, \dots, p_d > 0$. Then

$$\left| u - \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}} \right| \leq$$
$$\sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\} \\ \subset \{1,\dots,d\}}} \sum_{\underline{j} \in J_{e_1,\dots,e_k}} |C_{e_1,\dots,e_k}(h_{j_1}, \dots, h_{j_k})| h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} \left| \sum_{\underline{l} \in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}} \right|\,.$$

Proof. As $\sum_{\underline{i} \in J} c_{\underline{i}} = 1$ we have

$$u - \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}} = \sum_{\underline{i} \in J} c_{\underline{i}} (u - u_{\underline{i}})\,.$$

We may now substitute in the error splitting into the right hand side which leads to

$$\begin{aligned} \left| u - \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}} \right| &= \left| \sum_{\underline{i} \in J} c_{\underline{i}} \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\} \\ \subset \{1,\dots,d\}}} C_{e_1,\dots,e_k}(h_{i_{e_1}}, \dots, h_{i_{e_k}}) h_{i_{e_1}}^{p_{e_1}} \cdots h_{i_{e_k}}^{p_{e_k}} \right| \\ &\leq \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\} \\ \subset \{1,\dots,d\}}} \left| \sum_{\underline{i} \in J} c_{\underline{i}} C_{e_1,\dots,e_k}(h_{i_{e_1}}, \dots, h_{i_{e_k}}) h_{i_{e_1}}^{p_{e_1}} \cdots h_{i_{e_k}}^{p_{e_k}} \right|\,. \end{aligned}$$

For the inner most sum we use the decomposition (4.20) to obtain

$$\begin{aligned}
&\left|u - \sum_{\underline{i}\in J} c_{\underline{i}} u_{\underline{i}}\right| \\
&\quad \le \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\}\\ \subset\{1,\dots,d\}}} \left| \sum_{\underline{j}\in J_{e_1,\dots,e_k}} \sum_{\underline{l}\in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}} C_{e_1,\dots,e_k}(h_{l_{e_1}},\dots,h_{l_{e_k}}) h_{l_{e_1}}^{p_{e_1}} \cdots h_{l_{e_k}}^{p_{e_k}} \right| \\
&\quad \le \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\}\\ \subset\{1,\dots,d\}}} \sum_{\underline{j}\in J_{e_1,\dots,e_k}} |C_{e_1,\dots,e_k}(h_{j_1},\dots,h_{j_k})| h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} \left| \sum_{\underline{l}\in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}} \right|
\end{aligned}$$

as required. □

Corollary 4.51. *Let $u \in C([0,1]^d)$ and $J \subset \mathbb{N}^d$ be finite and non-empty. For each $\underline{i} \in J$ let $c_{\underline{i}} \in \mathbb{R}$ such that $\sum_{\underline{i}\in J} c_{\underline{i}} = 1$ and let each $u_{\underline{i}}$ be an approximation of u satisfying the extended point-wise error splitting model* (4.12) *with $q_1 > p_1 > 0, \dots, q_d > p_d > 0$, then*

$$\begin{aligned}
\left|u - \sum_{\underline{i}\in J} c_{\underline{i}} u_{\underline{i}}\right| &\le \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\}\\ \subset\{1,\dots,d\}}} |C_{e_1,\dots,e_k}| \left| \sum_{\underline{j}\in J} h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} c_{\underline{j}} \right| \\
&\quad + \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\}\\ \subset\{1,\dots,d\}}} \sum_{\underline{j}\in J_{e_1,\dots,e_k}} |D_{e_1,\dots,e_k}(h_{j_1},\dots,h_{j_k})| h_{j_1}^{q_{e_1}} \cdots h_{j_k}^{q_{e_k}} \left| \sum_{\underline{l}\in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}} \right| .
\end{aligned}$$

Proof. This follows the same proof as Proposition 4.50 for the two different components of the extended error splitting. As the $C_{e_1,\dots,e_k}$ in the leading component do not depend on the $h_{j_1},\dots,h_{j_d}$ they can be moved to the left of the sum over $\underline{j} \in J$. □

Now we may formulate a minimisation problem such that for a given $J \subset \mathbb{N}^d$ one finds coefficients which minimise the bounds of Proposition 4.50 or Corollary 4.51 depending on the error splitting model that one assumes. With the extended error splitting model used in the Corollary, if the coefficients are such that the $\sum_{\underline{j}\in J} h_{j_{e_1}}^{p_{e_1}} \cdots h_{j_{e_k}}^{p_{e_k}} c_{\underline{j}}$ terms are zero then the $C_{e_1,\dots,e_k}$ disappear and one obtains a higher order estimate. We first look at the problem of minimising the bound of Proposition 4.50 in detail.

Let $\boldsymbol{c} = \{c_{\underline{i}}\}_{\underline{i}\in J}$, then consider the minimisation of

$$\mathcal{J}_4(\boldsymbol{c}) := \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\}\\ \subset\{1,\dots,d\}}} \sum_{\underline{j}\in J_{e_1,\dots,e_k}} |C_{e_1,\dots,e_k}(h_{j_1},\dots,h_{j_k})| h_{j_{e_1}}^{p_{e_1}} \cdots h_{j_{e_k}}^{p_{e_k}} \left| \sum_{\underline{l}\in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}} \right| \tag{4.21}$$

subject to the equality constraint $\sum_{\underline{i}\in J} c_{\underline{i}} = 1$. The equality constraint here ensures consistency of the combination coefficients. This is essentially a weighted L_1 minimisation problems over the $N = \sum_{k=1}^{d} \sum_{\{e_1,\dots,e_k\}\subset\{1,\dots,d\}} |J_{e_1,\dots,e_k}|$ terms of the form $\left|\sum_{\underline{l}\in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}}\right|$ with weights $|C_{e_1,\dots,e_k}(h_{j_1},\dots,h_{j_k})| h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}}$. We can write this generically as the problem

$$\begin{aligned} &\text{minimise} && \|W\boldsymbol{c}\|_1 \\ &\text{subject to} && \mathbf{1}^{\mathsf{T}}\mathbf{c} = 1\,, \end{aligned} \tag{4.22}$$

where $\boldsymbol{c}$ has length $|J|$ and W is a $|J| \times N$ matrix mapping $\boldsymbol{c}$ to the each of the $\sum_{\underline{l}\in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}}$ terms (multiplied by their respective weights). This may be solved via the equivalent linear programming problem

$$\begin{aligned} &\text{minimise} && \mathbf{1}^{\mathsf{T}}\boldsymbol{d} \\ &\text{subject to} && \begin{bmatrix} W & -I \\ -W & -I \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} \le \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix} \\ &&& \mathbf{1}^{\mathsf{T}}\boldsymbol{c} = 1\,, \end{aligned}$$

where I is an $N \times N$ identity matrix and $\boldsymbol{d}$ is an N-vector (and $\mathbf{1}^{\mathsf{T}}$ has length N and $|J|$ in the first and second instance respectively).

One difficulty is in the estimation of the $|C_{e_1,\dots,e_k}(h_{j_1},\dots,h_{j_k})|$ which weight each term in the minimisation problem. Observe that they need only be estimated up to a constant factor, that is we need only know the relative size of the weights. In the classical sparse grid error bounds we use a constant $K > 0$ to bound all of these terms from above. If we take this approach in determining the coefficients than the value of K may be factored out and we need only minimise

$$\mathcal{J}_5(\boldsymbol{c}) := \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\}\\ \subset\{1,\dots,d\}}} \sum_{\underline{j}\in J_{e_1,\dots,e_k}} h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} \left| \sum_{\underline{l}\in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}} \right| . \tag{4.23}$$

However, if we can tighten the bound by treating the terms separately then one would expect the coefficients obtained from the minimisation problem should lead to a better result.

The estimation of the $|C_{e_1,\dots,e_k}(h_{j_1},\dots,h_{j_k})|$ terms is an interesting problem in itself. These values depend on both the exact u we are trying to approximate and the numerical scheme used to approximate u. In the absence of a detailed error analysis which tells us exactly what these terms are we can estimate them from some computed $u_{\underline{i}}$. Rather than have individual estimates for each $|C_{e_1,\dots,e_k}(h_{j_1},\dots,h_{j_k})|$ it is simpler to find $K_{e_1,\dots,e_k}$ for each $\{e_1,\dots,e_k\} \subset \{1,\dots,d\}$ such that $|C_{e_1,\dots,e_k}(h_{j_1},\dots,h_{j_k})| \leq K_{e_1,\dots,e_k}$ for all $h_{j_1},\dots,h_{j_k}$ (and $\boldsymbol{x}$ in the domain). For example, given a two dimensional problem with $p_1 = p_2 = p$ we note that

$$\begin{aligned} u_{(i_1,i_2)} - u_{(i_1-1,i_2)} &= (u - u_{(i_1-1,i_2)}) - (u - u_{(i_1,i_2)}) \\ &= C_1(h_{i_1-1})h_{i_1-1}^p - C_1(h_{i_1})h_{i_1}^p \\ &\quad + C_{1,2}(h_{i_1-1},h_{i_2})h_{i_1-1}^p h_{i_2}^p - C_{1,2}(h_{i_1},h_{i_2})h_{i_1}^p h_{i_2}^p \,. \end{aligned}$$

By increasing i_2 such that the contribution from $C_{1,2}$ terms are negligible and assuming that $C_1(h_{i_1}) \approx C_1(h_{i_1-1})$ for sufficiently large i_1 then one is able to estimate C_1. C_2 may be estimated in a similar fashion and $C_{1,2}$ may be estimated via

$$u_{(i_1,i_2)} - u_{(i_1-1,i_2)} - u_{(i_1,i_2-1)} + u_{(i_1-1,i_2-1)} \,.$$

In higher dimensions one estimates a $C_{e_1,\dots,e_k}$ term for $\{e_1,\dots,e_k\} \subset \{1,\dots,d\}$ by studying

$$\sum_{\left\{\underline{j}\in\{0,1\}^d : j_s=0 \text{ if } s\notin\{e_1,\dots,e_k\}\right\}} (-1)^{|\underline{j}|} u_{\underline{i}+\underline{j}} \,.$$

There are times when the elements of $\underline{i}$ may not be sufficiently large to accurately estimate the $C_{e_1,\dots,e_k}$ terms, for example in high dimensions where it is too costly to compute $\underline{i}$ without some of the components being small. In this thesis we typically stick with the case of generically bounding all of these terms by a single K and thus minimise (4.23). In practice we have observed that this minimisation problem often leads to combination coefficients that are the same as those produced by the GCP. An interesting direction to extend this work would be to apply techniques from uncertainty quantification to investigate the sensitivity of the coefficients obtained depending on the accuracy of the weights $C_{e_1,\dots,e_k}$ used.

Now consider $u_{\underline{i}}$ which satisfy the general extended error splitting (4.12). We observe that the minimisation over the bound for the remainder terms has identical structure to the minimisation (4.21) (just change the exponents from $p_1,\dots,p_d$ to $q_1,\dots,q_d$ and relabel the C as D). However, in addition to these terms we have an additional $2^d - 1$ terms to minimise, namely

$$|C_{e_1,\dots,e_k}| \left| \sum_{\underline{j} \in J} h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} c_{\underline{j}} \right|$$

for each $k = 1, \dots, d$ and $\{e_1, \dots, e_k\} \subset \{1, \dots, d\}$. This leads to the optimisation problem of minimising

$$\begin{aligned} \mathcal{J}_6(\boldsymbol{c}) := & \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\} \\ \subset\{1,\dots,d\}}} |C_{e_1,\dots,e_k}| \left| \sum_{\underline{j} \in J} h_{j_1}^{p_{e_1}} \cdots h_{j_k}^{p_{e_k}} c_{\underline{j}} \right| \\ & + \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\} \\ \subset\{1,\dots,d\}}} \sum_{\underline{j} \in J_{e_1,\dots,e_k}} |D_{e_1,\dots,e_k}(h_{j_1}, \dots, h_{j_k})| h_{j_{e_1}}^{q_{e_1}} \cdots h_{j_{e_k}}^{q_{e_k}} \left| \sum_{\underline{l} \in J_{\underline{j}|e_1,\dots,e_k}} c_{\underline{l}} \right| \end{aligned} \tag{4.24}$$

subject to the equality constraint $\sum_{\underline{i} \in J} c_{\underline{i}} = 1$. The estimation of the constants $C_{e_1,\dots,e_k}$ is done in the same way as before. The $D_{e_1,\dots,e_k}$ are more difficult requiring one to first compute the extrapolations $\tilde{u}_{\underline{i}}$ from Section 4.3 to eliminate the $C_{e_1,\dots,e_k}$ terms and then then compare the $\tilde{u}_{\underline{i}}$ for neighbouring $\underline{i}$ as was done for the $C_{e_1,\dots,e_k}$. As before, one can take a generic approach of bounding all of these terms by the same K in order to simplify the problem (that is K bounds both the $C_{e_1,\dots,e_k}$ and $D_{e_1,\dots,e_k}$). As the primary purpose of considering the extended error splitting is to obtain coefficients that result in an extrapolation it may be favourable use two constants K_C, K_D bounding the C and D terms respectively. By making the K_C large enough we can ensure that the minimisation problem will emphasise the cancellation of the $C_{e_1,\dots,e_k}$.

In practice we find that in some circumstances the extrapolation works and good results may be obtained. However there are times when the extrapolation does not work and the result is poor, in particular, worse than the results of the GCP. Thus we conclude that whilst this has potential to provide more accurate results it is less robust. It is possible that the robustness could be improved with careful tuning of the bounds on the C and D terms but we do not study this in more detail here.

4.5 Numerical Results

We provide some numerical results for advection in 2 spatial dimensions to demonstrate the extrapolation combination introduced in Section 4.3, and the combinations over arbitrary index sets developed in Section 4.4. Some of the numerical results presented in this section were published in the paper [65].

Consider the index sets $J_{n,\tau,l} \subset \mathbb{N}^2$ given by

$$J_{n,\tau,l} := \{\underline{i} \in \mathbb{N}^2 : i_1, i_2 \geq \tau \text{ and } n - l < |\underline{i}| \leq n\} ,$$

with n, τ and l being the level, truncation and layer count parameters respectively. We require $n \geq 2\tau$ for this set to be non-empty. The classical/truncated combination coefficients for $i \in J_{n,\tau,l}$ with $l \geq 2$ are

$$c_{\underline{i}} = \begin{cases} 1 & \text{if } |\underline{i}| = n \\ -1 & \text{if } |\underline{i}| = n - 1 , \end{cases} \tag{4.25}$$

which correspond to the truncated combination $u^t_{n-2\tau,\underline{\tau}}$ from Section 4.1. Note that coefficients which are not specified are zero. For these index sets with $l \geq 2$ the GCP has a unique solution which leads to exactly the truncated combination coefficients given. These combinations will be compared with those using c_i derived from the minimisation of the order p error splitting estimate (4.21). We will also compare with the combination of multi-variate extrapolations for second order schemes, namely $\tilde{u}_{\underline{i}}$ as defined in Section 4.3. For $i \in J_{n,\tau,l}$ with $l \geq 4$ and $n \geq 2(\tau + 2)$ the coefficients are

$$c_{\underline{i}} = \begin{cases} \frac{16}{9} & \text{if } |\underline{i}| = n \text{ and } \underline{i} \geq (\tau+1, \tau+1) \\ \frac{-24}{9} & \text{if } |\underline{i}| = n-1 \text{ and } \underline{i} \geq (\tau+1, \tau+1) \\ \frac{-4}{9} & \text{if } |\underline{i}| = n-1 \text{ and } \underline{i} \ngeq (\tau+1, \tau+1) \\ 1 & \text{if } |\underline{i}| = n-2 \text{ and } \underline{i} \geq (\tau+1, \tau+1) \\ \frac{5}{9} & \text{if } |\underline{i}| = n-2 \text{ and } \underline{i} \ngeq (\tau+1, \tau+1) \\ \frac{-1}{9} & \text{if } |\underline{i}| = n-3 . \end{cases} \tag{4.26}$$

These results are compared with c_i derived from the minimisation of the order p, q error splitting estimate (4.24) as both are expected to give higher order approximations. We will also compare results of the GCP and the two error splitting combinations for several different randomly chosen subsets $J \subset J_{16,4,9}$ with $\mathrm{E}[|J|] = 0.8|J_{16,4,9}|$. Figure 4.6 shows the combination coefficients derived from the GCP for one such subset of $J_{16,4,9}$.

To summarise, in the results that follow we refer to the following combinations:

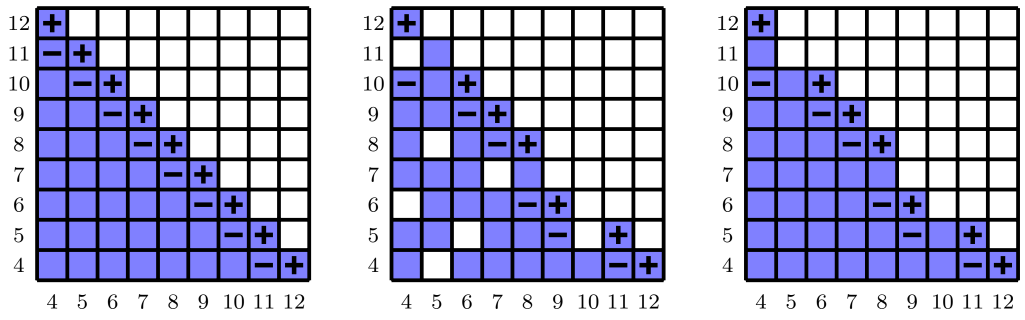


Figure 4.6: *On the left we depict the set* $J_{16,4,9}$ *(shaded blue) with the classical combination coefficients (+ for* $+1$ *and* $-$ *for* -1*). In the middle we depict a subset* J *of* $J_{16,4,9}$ *with the corresponding combination coefficients obtained from the* GCP*. On the right we depict the downset* $I \subset J{\downarrow}$ *which corresponds to the* GCP *solution. Note that* I *also includes* $\{(3,12),(12,3)\}{\downarrow}$ *which are not visible.*

- The GCP combination has corresponding hierarchical coefficients which minimise (4.17) subject to the appropriate constraints. For the sets $J_{n,\tau,l}$ this is exactly the coefficients given by (4.25).
- The *extrapolation* combination for the sets $J_{n,\tau,l}$ has coefficients as in (4.26).
- The *order p error splitting* combination has coefficients which minimise (4.21) subject to the consistency constraint.
- The *order p, q error splitting* combination has coefficients which minimise (4.24) subject to the consistency constraint.

For the latter two we use the estimate $|C_{e_1,\dots,e_k}| = 1$ and $|D_{e_1,\dots,e_k}| = 1$ unless stated otherwise.

4.5.1 2D advection problem with constant flow field

Here we perform tests on numerical solutions of the scalar advection equation

$$\frac{\partial u}{\partial t} + \boldsymbol{a} \cdot \nabla u = 0\,, \tag{4.27}$$

where $u : [0,1]^2 \to \mathbb{R}$ and $\boldsymbol{a} = (1,1)$. We specify an initial condition at $t = 0$ given by $u_0 = \cos(2\pi x)\sin(2\pi y)$ and enforce periodic boundary conditions. We evolve up to $t = 0.25$ using second order centred finite difference discretisation of spatial derivatives and the classical fourth order Runge–Kutta scheme for integration over time (thus $p = 2$ and $q = 4$ in the error splitting models). In particular,

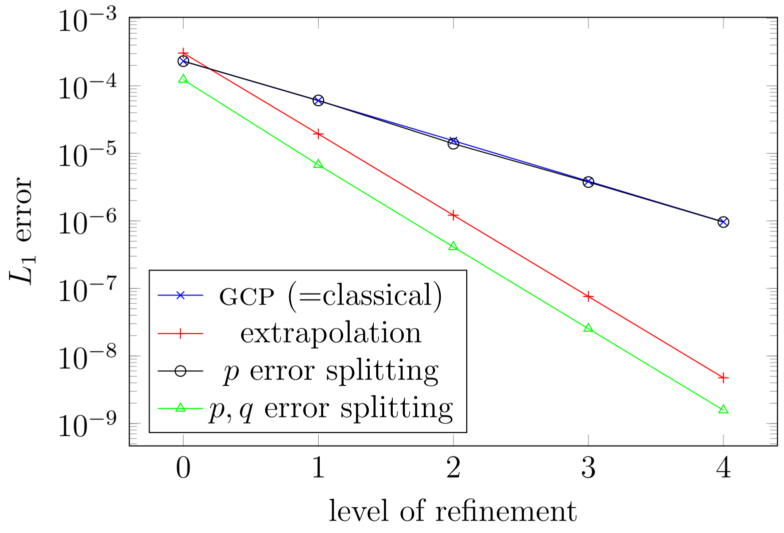


Figure 4.7: *Starting with* $J_{12,4,4}$ *we compare the classical combination* (4.25)*, the extrapolation combination* (4.26)*, the order* p *error splitting combination (minimising* (4.21)*) and the order* p,q *error splitting combination (minimising* (4.24)*). The grids of* $J_{12,4,4}$ *are refined in both spatial dimensions several times and the computations repeated. Note that the results for* GCP *and order* p *error splitting are overlapping.*

given the numerical approximations $U^n_{i,j} \approx u(t_n, x_i, y_j) = u(n\Delta t, i\Delta x, j\Delta y)$ we use the numerical scheme

$$U^{n+1}_{i,j} = U^n_{i,j} + \frac{\Delta t}{6}(k^{(1)}_{i,j} + 2k^{(2)}_{i,j} + 2k^{(3)}_{i,j} + k^{(4)}_{i,j})$$

where

$$\begin{aligned}
k^{(1)}_{i,j} &= -\frac{a_1}{2\Delta x}(U^n_{i+1,j} - U^n_{i-1,j}) - \frac{a_2}{2\Delta y}(U^n_{i,j+1} - U^n_{i,j-1}) \\
k^{(2)}_{i,j} &= k^{(1)}_{i,j} - \frac{a_1\Delta t}{4\Delta x}(k^{(1)}_{i+1,j} - k^{(1)}_{i-1,j}) - \frac{a_2\Delta t}{4\Delta y}(k^{(1)}_{i,j+1} - k^{(1)}_{i,j-1}) \\
k^{(3)}_{i,j} &= k^{(1)}_{i,j} - \frac{a_1\Delta t}{4\Delta x}(k^{(2)}_{i+1,j} - k^{(2)}_{i-1,j}) - \frac{a_2\Delta t}{4\Delta y}(k^{(2)}_{i,j+1} - k^{(2)}_{i,j-1}) \\
k^{(4)}_{i,j} &= k^{(1)}_{i,j} - \frac{a_1\Delta t}{2\Delta x}(k^{(3)}_{i+1,j} - k^{(3)}_{i-1,j}) - \frac{a_2\Delta t}{2\Delta y}(k^{(3)}_{i,j+1} - k^{(3)}_{i,j-1})\,.
\end{aligned}$$

In Figure 4.7 we compare the rate of convergence of several methods starting with the index set $J_{12,4,4}$ and then refining each grid uniformly by a factor of 2 for subsequent computations (corresponding to index sets $J_{14,5,4}$, $J_{16,6,4}$, $J_{18,7,4}$ and $J_{20,8,4}$). We observe that the classical combination and the order p error splitting result have the same order of convergence (2) and have very similar results in general. The combination of extrapolations and the order p,q error splitting

Table 4.1: *Here we give the L_1 error for combinations obtained via interpolation (*GCP*) and error splitting ('p split' and 'p, q split') estimates for the combination error over 20 random samples $J \subset J_{16,4,9}$ with each element having an 80% chance of appearing in J. The 10 samples on the left use the generic weighting $|C_{e_1,\dots,e_k}| = 1$ whilst the 10 samples on the right use a rough estimate of $\|C_{e_1,\dots,e_k}\|_\infty$ which helps to stabilise the order p error splitting results. Note that* GCP *solutions do not depend on the $C_{e_1,\dots,e_k}$ estimate.*

Unit weighting				$\|C_{e_1,\dots,e_k}\|_\infty$ estimate			
sample	GCP	p split	p,q split	sample	GCP	p split	p,q split
1	2.636E-6	3.680E-6	7.893E-7	11	1.457E-6	1.457E-6	1.275E-6
2	1.312E-6	3.103E-5	1.452E-6	12	2.613E-6	2.614E-6	1.267E-6
3	3.340E-6	7.946E-6	4.680E-6	13	1.153E-6	1.153E-6	3.655E-7
4	1.153E-6	3.680E-6	7.223E-7	14	1.153E-6	1.153E-6	3.655E-7
5	3.253E-6	3.564E-6	7.866E-7	15	1.488E-6	1.489E-6	2.349E-7
6	1.932E-6	2.968E-5	1.554E-6	16	1.455E-6	1.312E-6	3.655E-7
7	1.312E-6	7.946E-6	4.113E-7	17	2.585E-6	1.153E-6	3.655E-7
8	2.590E-6	3.564E-6	3.683E-6	18	1.153E-6	1.153E-6	2.795E-5
9	2.636E-6	1.260E-5	5.361E-7	19	1.801E-6	1.630E-6	2.359E-7
10	1.631E-6	3.103E-5	7.866E-7	20	2.822E-6	2.822E-6	1.723E-6
mean	2.180E-6	1.347E-5	1.540E-6	mean	1.768E-6	1.594E-6	3.415E-6
stdev	8.170E-7	1.215E-5	1.457E-6	stdev	6.584E-7	6.182E-7	8.637E-6

both exhibit a higher rate of convergence (4) and the order p, q error splitting result outperforms the combination of extrapolations by a factor of approximately 2. A generic weighting of 1 is used for the $|C_{e_1,\dots,e_k}|$ and $|D_{e_1,\dots,e_k}|$ in these tests.

In Table 4.1 we compare the error splitting based coefficients with solutions of the GCP. We take random samples $J \subset J_{16,4,9}$ with each multi-index in $J_{16,4,9}$ appearing in J with probability 0.8. We then compute coefficients using the different approaches developed in Section 4.4 and compare the resulting combinations. On the left the first 10 samples use the generic weighting $|C_{e_1,\dots,e_k}| = 1$ for the error splitting approach (similar for the D). We see that the GCP outperforms the order p error splitting coefficients (p split). The order p, q error splitting coefficients (p, q split) have higher order convergence and thus outperforms the order p splitting results in most cases but only outperforms the the GCP approach in 7/10 cases. On the right we have an additional 10 samples where the $C_{e_1,\dots,e_k}$ are weighted with a rough estimate of $\|C_{e_1,\dots,e_k}\|_\infty$ (and similarly for D) for the error splitting approaches. This leads to significant improvement in the order p error splitting results results which typically yields similar combination coefficients to the GCP and even outperforms in some cases. The order p, q error splitting results also improve and outperform the other approaches with the exception of one out-

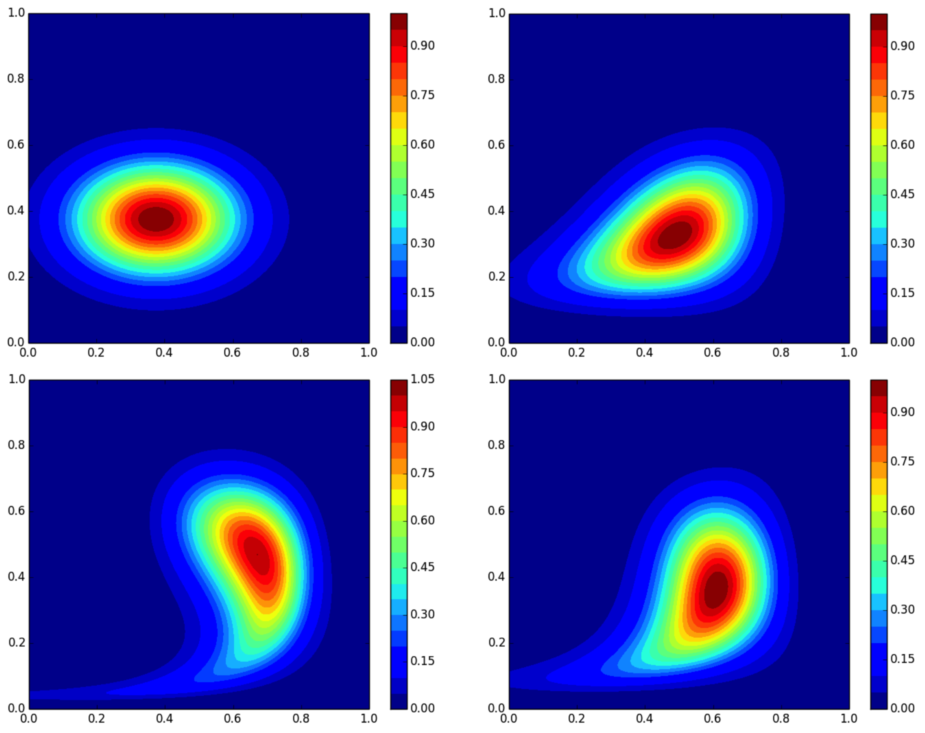


Figure 4.8: *The evolution of the initial condition under the divergence free velocity field* (4.28)*. In a clockwise direction from the top left we have the solution at times* 0*,* 0.25*,* 0.5 *and* 0.75 *respectively.*

lier. We conclude that the error splitting based coefficients can outperform the GCP approach when an error splitting is applicable and the $C_{e_1,\dots,e_k}$ are estimated with reasonable accuracy. Without this estimate of $C_{e_1,\dots,e_k}$ the error splitting results are have significant variance and appear to be less robust than the those obtained with the GCP.

4.5.2 2D advection problem with divergence free flow field

Here we again solve a 2D advection equation (4.27) but this time with $\boldsymbol{a}$ depending on the spatial coordinates. In particular, consider the divergence free velocity field

$$\boldsymbol{a}(x,y) = (\sin(\pi x)\cos(\pi y), -\cos(\pi x)\sin(\pi y)) \tag{4.28}$$

for $(x,y) \in [0,1]^2$ and the solutions of the advection equation $\frac{\partial u}{\partial t} + \boldsymbol{a} \cdot \nabla u = 0$ with initial condition $u(\boldsymbol{x},0) = \exp(-6 + 4(1-\pi^2(x-3/8)^2) + 2(1-2\pi^2(y-3/8)^2))$. The initial condition is a Gaussian peak centred at $(3/8, 3/8)$ which follows the velocity field around the centre of the domain in an anti-clockwise direction time evolves as depicted in Figure 4.8. Note that along the boundary the velocity

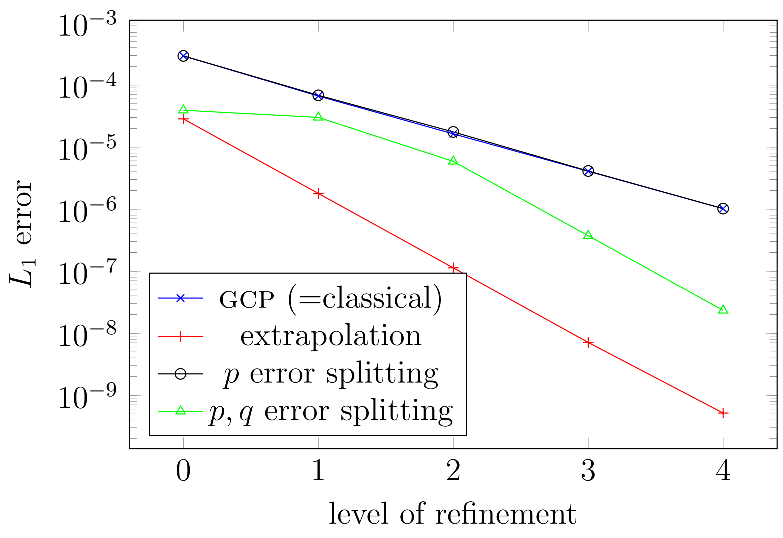


Figure 4.9: *Starting with $J_{12,4,4}$ we compare the four different combinations (*GCP*, extrapolation, order p error splitting and order p, q error splitting combinations) for the rotating velocity field* (4.28)*. The grids of $J_{12,4,4}$ are refined in both spatial dimensions several times and the computations repeated.*

field is perpendicular to the boundary normal such that there is zero flux leaving or entering the domain. We solve up to time $t = 0.25$ and perform the same experiments as was done for the constant velocity field. The exact solution is obtained by using a high order ODE solver to integrate backwards along the velocity field and L_1 error is computed relative to this.

Figure 4.9 shows analogous results to those of Figure 4.7 for this rotating velocity field and initial condition. As before we compare the rate of convergence of the different methods by starting with the index set $J_{12,4,4}$ and then refining each grid uniformly by a factor of 2 for subsequent computations. A generic weighting of 1 is again used for the $|C_{e_1,\dots,e_k}|$ and $|D_{e_1,\dots,e_k}|$ in these tests when determining the order $p/p, q$ error splitting combinations. The classical combination and the order p error splitting combination are both very similar again and exhibit 2nd order convergence. The combination of extrapolations exhibits 4th order convergence. However, the result from the p, q error splitting is not as accurate for this problem. For the first two refinements the error does not change substantially, and it is only for the subsequent two refinements that 4th order convergence is achieved.

In Table 4.2 we compare the error splitting based coefficients with solutions of the GCP for this new problem analogous to the results of Table 4.1. As before we take random samples of $J \subset J_{16,4,9}$ with each multi-index in $J_{16,4,9}$ appearing

Table 4.2: *Here we give the L_1 error for combinations obtained via interpolation (gcp) and error splitting ('p split' and 'p, q split') estimates for the combination error over 20 random samples $J \subset J_{16,4,9}$ with each element having an 80% chance of appearing in J. The 10 samples on the left use the generic weighting $|C_{e_1,\dots,e_k}| = 1$ whilst the 10 samples on the right use a rough estimate of $\|C_{e_1,\dots,e_k}\|_\infty$. Note that* GCP *solutions do not depend on the $C_{e_1,\dots,e_k}$ estimate.*

Unit weighting				$\|C_{e_1,\dots,e_k}\|_\infty$ estimate			
sample	GCP	p split	p, q split	sample	GCP	p split	p, q split
1	5.6031E-6	1.4712E-6	6.7669E-6	11	2.5795E-6	2.5794E-6	1.9397E-5
2	4.0978E-6	7.0190E-5	1.3552E-5	12	5.5722E-6	5.5722E-6	7.0050E-7
3	1.7608E-6	6.8661E-6	6.7584E-6	13	8.9001E-6	8.9001E-6	1.0675E-5
4	4.3890E-6	2.3201E-5	6.7565E-6	14	4.4020E-6	4.3760E-6	7.9408E-6
5	1.7096E-6	1.2359E-5	6.6645E-6	15	4.5229E-6	4.5229E-6	1.8274E-6
6	2.4972E-6	6.6887E-6	6.7590E-6	16	2.8716E-6	2.8640E-6	5.0656E-7
7	1.7096E-6	6.5493E-6	6.4503E-6	17	4.3207E-6	4.3207E-6	7.1996E-6
8	1.3830E-5	1.5578E-5	6.7565E-6	18	4.2122E-6	4.2122E-6	3.5387E-6
9	2.1760E-6	1.6241E-5	6.7590E-6	19	5.5297E-6	5.5297E-6	1.5854E-5
10	1.4610E-5	1.8233E-5	1.7650E-5	20	2.1876E-6	2.1876E-6	1.4133E-6
mean	5.2383E-6	1.7738E-5	8.4873E-6	mean	4.5098E-6	4.5065E-6	6.9053E-6
stdev	4.6657E-6	1.8567E-5	3.6741E-6	stdev	1.8271E-6	1.8279E-6	6.3104E-6

in J with probability 0.8 and then compare the combinations obtained via the different approaches. On the left the first 10 samples use the generic weighting $|C_{e_1,\dots,e_k}| = 1$ for the error splitting approach (similar for the D). The GCP outperforms the order p error splitting coefficients (p split) in most cases and is much more robust having a lower standard deviation. However, the order p, q error splitting coefficients (p, q split) does not perform significantly better than the order p error splitting combination except where the latter performs particularly poorly. Curiously these 10 order p, q error splitting combinations have lower standard deviation than the GCP combinations for this particular problem although the GCP combinations have a smaller mean. On the right we have an additional 10 samples where the $C_{e_1,\dots,e_k}$ are weighted with a rough estimate of $\|C_{e_1,\dots,e_k}\|_\infty$ (and similarly for D) for the error splitting approaches. This leads to significant improvement in the order p error splitting results which typically the same result as the GCP and is slightly smaller in the coupe of instances in which they differ. The mean error of the order p, q error splitting combinations also improves although the standard deviation increases. In 5/10 of these results we observe that the order p, q error splitting combination is the best of the three but is significantly worse on the other occasions. With a better estimate of the

$C_{e_1,\ldots,e_k}$ and D terms it may be possible to improve the success rate of these order p, q error splitting combinations. We conclude that the GCP is again the more reliable approach and produces good results without the need for fine tuning.

Chapter 5

Fault Tolerant Combination Technique

This chapter describes and analyses a fault tolerant adaptation of the combination technique based upon the generalisations developed in Chapter 4. This work is related to several publications which develop and demonstrate the fault tolerant combination technique (FTCT) and its application to high performance computing [67, 68, 66, 69, 5, 120, 3, 79]. In Section 5.1 two checkpointing routines for the combination technique are described. Given that checkpoint-restart is the most common approach to fault tolerance used in the computing community today this will be used as a basis of comparison for other methods of algorithm based fault tolerance which will be developed. In Section 5.2 we describe an implementation of the FTCT based on the work in Section 4.4. This form of algorithm based fault recovery is lossy in the sense that data is not recovered exactly. Rather, we adjust our combination to obtain reasonable results with the data that remains. The end result is therefore stochastic in nature depending on how many faults occurred and which component solutions were affected. As such, we analyse the expected error by utilising the fault models developed in Chapter 1 to extend some of the error estimates presented in Chapter 2 to the FTCT. Section 5.2.4 presents some numerical results for the FTCT applied to the advection equation with simulated faults. The results demonstrate that the overhead is low and that the expected error is close to the error of the solution obtained in the absence of faults. We end with Section 5.3 which consists of many additional comments and remarks in relation to fault tolerant computations with the combination technique.

5.1 Checkpointing the Combination Technique

Checkpoint restart has historically been the most widely used approach for hard fault recovery in high performance computing. As such we compare the overheads of our new approach with those of checkpoint restart. It is important to point out that there are many varieties of checkpoint restart and that there has been extensive research in recent years to improve the performance of checkpoint restart for peta/exa scale computing. However, as we clearly are not able to investigate and compare against all of these varieties we look at two of relatively simple checkpoint restart implementations which we refer to as 'global' and 'local' checkpointing.

The 'global' checkpointing is based on a traditional checkpointing fault model. In this model is assumed that any fault, independent of nature and origin, causes the entire application to stop immediately. To overcome this the complete state of the application is saved to stable storage periodically. After a failure has occurred the application is restarted. From here the last saved state is reloaded into memory and computations continue from this state. Because the entire state of the application must be saved checkpoints are typically large. The total time required to take a checkpoint is typically limited by the write speed of the storage system. Historically hard disks in a separate storage cluster were used whilst in recent years systems have begun to used solid state disks attached to each node of the cluster.

In Algorithm 1 an outline of 'global' checkpointing implemented with the combination technique is described. On starting, the application either starts from the initial condition or the most recent (complete) checkpoint. It then begins the main computation loop in which the state of the application is saved at the end of each iteration. In the context of the combination technique the state of the application is the current field on the combined sparse grid which is equivalent to the current field of all of the coarse solutions. Although not explicit in the pseudocode we assume that various run time parameters like the current iteration are also saved. The main loop runs until either it reaches the end of the last iteration or it is terminated due to a failure. It is also assumed that checkpoints are written in a redundant fashion such that if a failure results in an incomplete checkpoint then the previous completed checkpoint still exists and can thus be used for restart. Note that one may modify the algorithm to evolve the coarse solutions several times before combining with checkpoints occurring after each evolution. This allows the checkpoint interval to be decoupled from

Algorithm 1 Outline of 'global' checkpointing algorithm for the combination technique. Here it is assumed that a failure results in immediate termination of the application/algorithm which is then restarted from the beginning.

```
if no checkpoint exists then
    set coarse solutions to initial condition and set $n = 0$
else
    load coarse solutions and current iteration $n$ from checkpoint
end if
while $n < N$ do
    evolve each coarse solution by some fixed number of time steps
    hierarchise each coarse solution
    combine hierarchical surpluses // communication between processes
    reconstruct coarse solutions
    set $n = n + 1$
    checkpoint each coarse solution and the step counter $n$
end while
```

the combination interval (up to an integer multiple).

Remark 5.1. We pause for a moment to describe how the hierarchical surpluses are combined in our algorithms. Given approximations $u_{\underline{i}}$ for $\underline{i} \in I$ and a combination $u_I = \sum_{\underline{i} \in I} c_{\underline{i}} u_{\underline{i}}$ then the hierarchical implementation of the combination technique goes as follows.

- We first hierarchise each $u_{\underline{i}}$ for $\underline{i} \in I$. This means computing the coefficients of the hierarchical basis functions described in Section 2.1. Here we will denote $H_{\underline{j}}(u_{\underline{i}})$ to be the $\underline{j}$th hierarchical surplus of the approximation $u_{\underline{i}}$. Note that unlike interpolation problems each hierarchical surplus will generally differ for each approximation $u_{\underline{i}}$ (that is given $\underline{i} \neq \underline{k}$ and $\underline{j} \leq \underline{i}$ and $\underline{j} \leq \underline{k}$ then typically $H_{\underline{j}}(u_{\underline{i}}) \neq H_{\underline{j}}(u_{\underline{k}})$). Also observe that $H_{\underline{j}}(u_{\underline{i}}) = 0$ if $\underline{j} \not\leq \underline{i}$.
- Now we apply the combination formula for each of the hierarchical surpluses. That is, for each $\underline{j} \in I{\downarrow}$, we compute $H_{\underline{j}}(u_I) = \sum_{\underline{i} \in I} c_{\underline{i}} H_{\underline{j}}(u_{\underline{i}})$.
- One now reconstructs/updates each of the $u_{\underline{i}}$ as a sample of the combination u_I via the update $u_{\underline{i}} \leftarrow \sum_{\underline{j} \leq \underline{i}} H_{\underline{j}}(u_I)$.

The 'local' checkpointing approach is based on a fault model in which processes operate independently. That is, we assume that when a process fails the application is not terminated. The failed process is restarted (possibly on a different physical processor in the event of hardware failure) whilst all other processes continue uninterrupted. The only data lost is the most recent computation on the failed process. Computations which were previously completed on the failed process are restored from checkpoints made ‘locally’ by that process.

Algorithm 2 Outline of ‘local’ checkpointing algorithm for the combination technique. This parallel algorithm describes the procedure on each process. We assume a failure on one process requires the restart of only that particular process. The restarted process starts from the beginning and is able to continue computing from the correct state based on the counters n, m_p.

```
if no checkpoint exists then
    compute a load balancing of coarse solutions to processes
    get list of coarse solutions $u[0], \dots, u[M_p - 1]$ assigned to this process
    set $n = m_p = 0$ and coarse solutions $u[0], \dots, u[M_p - 1]$ to initial condition
    checkpoint load balancing, counters $n, m$ and solutions $u[0], \dots, u[M_p - 1]$
else
    load the load balancing information and counters $n, m_p$ from checkpoint
    load the $u[0], \dots, u[M_p - 1]$ assigned to this process from checkpoints
end if
while $n < N$ do
    for $i_p = m_p, \dots, M_p - 1$ do
        evolve coarse solution $u[i_p]$ by some fixed number of iterations
        checkpoint the coarse solution $u[i_p]$ and counter $m_p = i_p + 1$
    end for
    hierarchise coarse solutions $u[0], \dots, u[M_p - 1]$
    combine hierarchical surpluses // communication between processes
    reconstruct coarse solutions $u[0], \dots, u[M_p - 1]$
    checkpoint $u[0], \dots, u[M_p - 1]$ and the counters $m_p = 0$ and $n = n + 1$
end while
```

In Algorithm 2 an outline of ‘local’ checkpointing implemented with the combination technique is described. This is a somewhat simplified description and in practice one would only load data from checkpoints if there had been a recent failure on that process. Further, there would need to be a mechanism for the application to re-enter at the correct point in the computation which is not described in this pseudocode. The advantage of this procedure is that checkpoints can be saved independently by processes during the evolution of the coarse grids and that the entire application need not be restarted, only the affected process/processes. This model is representative of something which may be competitive with the modern and sophisticated checkpoint-restart implementations. Similar to the 'global' checkpointing algorithm, we may again evolve and checkpoint the component solutions several times prior to combination to decouple the checkpoint interval from the combination interval.

In our numerical experiments using fault simulation the implementation of these checkpointing procedures saves checkpoints to local memory rather than stable storage. We can do this because faults are only simulated, that is no process is actually killed. This gives the checkpoints a further advantage in our numerical simulations as the write speed to local memory is much higher than the read/write speeds in any real checkpoint-restart implementation that writes to stable storage. Thus our reporting of checkpoint restart overheads will be somewhat optimistic.

We wish to estimate the overhead of both checkpointing algorithms using the stochastic models developed in Chapter 1. In Section 1.4.2 we studied a renewal process model for checkpointing which we used to estimate the optimal checkpoint restart interval. This was effectively a model of the 'global' checkpointing algorithm. Recall that given the independent and identically distributed random variables X_i denoting the time between successive failures, c the time required to save a checkpoint, s the startup time of the application and r the computation time between checkpoints, then the waste time (i.e. time not spent on the core computation) between the $i-1$th and ith failures is

$$R_i = X_i - r\left\lfloor \frac{\max\{X_i - s, 0\}}{c+r} \right\rfloor .$$

From Proposition 1.20 we know that when the X_i are exponentially distributed then

$$\mathrm{E}[R_i] = \lambda - \frac{re^{-s/\lambda}}{e^{(c+r)/\lambda} - 1}$$

where $\lambda = \mathrm{E}[X_i]$. Further, we showed that a minimum for $\mathrm{E}[R_i]$ is achieved for $r + c \approx \sqrt{2c\lambda}$ when $r + c \ll \lambda$.

The relative overhead of the checkpointing algorithm is given by $\frac{T-T_{\mathrm{comp}}}{T_{\mathrm{comp}}}$ where T_{comp} is the computation time in the absence of checkpointing and faults and T is the computation time in the presence of checkpointing and faults. If the computation requires N computation cycles to complete, then the total computation time in the absence of checkpointing and faults is $T_{\mathrm{comp}} = s + Nr$. T

is stochastic in nature as it depends on when faults occur during the computation. As such we will estimate $\mathrm{E}[T]$ so that we may estimate the expected relative overhead. To simplify the calculation will will assume the startup time is negligible, that is $s = 0$ and thus $T_{\text{comp}} = Nr$. Consider a compute checkpoint cycle starting at some time t. The probability that this cycle completes successfully is $\Pr(N(t + r + c) - N(t) = 0)$, i.e. the probability that there are no renewals/faults in the interval $[t, r + c)$. Given a random starting time t this is equivalent to the probability that the forward recurrence time is larger than $r+c$, i.e. $\Pr(S_{N(t)+1} - t \geq r + c)$. If the time between failures is exponentially distributed then we know the forward recurrence is identically distributed to the X_i. Thus the probability of no failure during a randomly chosen compute checkpoint cycle is $\Pr(X_1 \geq r + c) = e^{-(r+c)/\lambda}$ where $\lambda = \mathrm{E}[X_1]$. Further, as the exponential distribution is memory-less the same then the probability of no failure is the same for all compute checkpoint cycles. We require N successes for the algorithm to complete with the last trial being a success. Given, $K \geq N$ trials then we want $N - 1$ successes out of $K - 1$ trials followed by a success. The probability of this event is $q\binom{K-1}{N-1}q^{N-1}(1 - q)^{K-N}$ where $q = \Pr(X_1 > r + c) = e^{-(r+c)/\lambda}$. It is straightforward to check that $\sum_{K=N}^{\infty} p\binom{K-1}{N-1}q^{N-1}(1 - q)^{K-N} = 1$ and that the expectation of the number of trials K which yields N successes (with the last a success) is

$$\sum_{K=N}^{\infty} Kq\binom{K-1}{N-1}q^{N-1}(1 - q)^{K-N} = \frac{N}{q} .$$

Thus the expected number of compute checkpoint cycles in which a failure occurs is $\frac{N}{q} - N = \frac{N(1-q)}{q}$. In each these compute checkpoint cycles the expected wasted time is given by

$$\mathrm{E}[X_1 \mid X_1 < r + c] = \int_0^{r+c} \frac{xe^{-x/\lambda})}{\lambda \Pr(X_1 < r + c)}\, dx = \lambda - \frac{(r + c)}{e^{(r+c)/\lambda} - 1} .$$

Now from Section 1.4.2 we know that for $r + c \ll \lambda$ we have $\frac{1}{e^{(r+c)/\lambda}-1} \approx \frac{\lambda}{r+c} - \frac{1}{2}$ and thus

$$\mathrm{E}[X_1 \mid X_1 < r + c] \approx \frac{r + c}{2} .$$

Therefore the expected lost time from a failed compute-checkpoint cycle is approximately $\frac{r+c}{2}$. Thus the expected computation time in the presence of checkpoints and failures is $\mathrm{E}[T] = N(r + c) + \frac{N(1-q)(r+c)}{2q}$. It follows that the expected relative overhead is

$$\mathrm{E}\left[\frac{T - T_{\text{comp}}}{T_{\text{comp}}}\right] = \frac{\mathrm{E}[T] - Nr}{Nr} \approx \frac{Nc + \frac{N(1-q)(r+c)}{2q}}{Nr} = \frac{c}{r} + \frac{(r + c)(1 - q)}{2rq} .$$

The $\frac{c}{r}$ term is the contribution of checkpoints of successful compute checkpoint cycles whilst the $\frac{(r+c)(1-q)}{2rq}$ term is the expected time spent on computations which are lost due to faults. Notice that the N drops out such that the expected relative overhead does not depend on the number of iterations. This is a consequence of our assumption that $s = 0$ and the memory-less property of the exponential distribution.

Under certain assumptions the 'local' checkpointing algorithm can also be modelled with the same renewal process. Suppose that the evolution of each $u[i_p]$ by one time step takes the same amount of time, the time to checkpoint each $u[i_p]$ is $c' = c/M_p$, and that the time spent outside the evolution loop is negligible. In these circumstances we effectively have the global checkpointing algorithm over $N \times M$ compute cycles of length r' with checkpoint time c'. Note that the optimal r' in this scenario is $r' \approx -c' + \sqrt{2c'M_p} = -c/M_p + \sqrt{2c\lambda/M_p}$. Additionally the probability of success in an interval of length $r' + c'$ is $q' = \Pr(X_1 > r' + c') = e^{-(r'+c')/\lambda}$. Thus the expected relative overhead is simply

$$\mathrm{E}\left[\frac{T - T_{\text{comp}}}{T_{\text{comp}}}\right] \approx \frac{c'}{r'} + \frac{(r' + c')(1 - q')}{2r'q'} \, .$$

As an example, suppose $\lambda = 10^4$, $s = 0$, $c = 10$ and $M_p = 100$. For 'global' checkpointing the optimal interval is approximately $r = -10 + \sqrt{2 \times 10^5} \approx 437.2$ which leads to an upper bound for the overhead of $\approx 4.63\%$. For 'local' checkpointing we obtain $r' = -0.1 + \sqrt{2 \times 10^3} \approx 44.62$ which leads to an upper bound for the overhead of $\approx 0.449\%$. Thus we see that the 'local' checkpointing algorithm is approximately one tenth of the overhead of the 'global' checkpointing algorithm. In general we expect the improvement to be a factor of $\approx M^{-1/2}$.

Remark 5.2. In the papers [67, 79, 80] it is discussed how the combination technique could be implemented within a map-reduce framework. The map-reduce framework typically involves one master process that delegates a large number of relatively small tasks to a collection of worker processes. A form of fault tolerance is achieved in many implementations by remapping tasks as necessary, for example, if a worker process is killed or fails to complete an assigned task within a specified time limit then the master re-sends that task to a different worker process. If the master is immediately notified of failures then the overheads in this model are comparable to the 'local' checkpointing discussed here and thus we do not provide a separate analysis for a map-reduce based combination technique.

5.2 Fault Tolerant Combination Technique

Suppose we have a set I of multi-indices for which we intend to compute each of the solutions $u_{\underline{i}}$ and combine according to

$$u_{\underline{i}} = \sum_{\underline{i} \in I} c_{\underline{i}} u_{\underline{i}} \tag{5.1}$$

where the $c_{\underline{i}}$ are given by a solution to the general coefficient problem (GCP), see Section 4.4. As each of the $u_{\underline{i}}$ can be computed independently the computation of these is easily distributed across different process pools in a high performance computer. To simplify the discussion we will assume these process pools consist of hardware nodes. Suppose that one or more of these nodes experiences a fault, which may be hardware or software in nature. As a result, some of our $u_{\underline{i}}$ may not have been computed correctly. We denote $J \subset I$ to be the set of indices for which the $u_{\underline{i}}$ where not correctly computed. A lossless approach to fault tolerance would be to recompute $u_{\underline{i}}$ for $\underline{i} \in J$, for example as described in algorithm 2. However, recomputation in a parallel environment is costly even if it is just for one $u_{\underline{i}}$. For example, consider 100 process pools of equal size with an equal balance of workload, and suppose that one of the process pools is delayed by time t as a result of recomputing a $u_{\underline{i}}$, then the other 99 process pools are idle making the parallel efficiency at most 1% for that period of time. To avoid this we propose a lossy approach to fault tolerance in which the failed solutions are not recomputed. This means we must find new combination coefficients $c_{\underline{i}}$ for $\underline{i} \in I \backslash J$. In Section 4.4 we discussed several approaches for finding combination coefficients given an arbitrary collection of grids. Any of those methods could be used to find new combination coefficients for $I \backslash J$. The numerical results in Section 4.5 indicated that the combinations obtained via the GCP were more robust than the other approaches. It also has the advantage of working well without the need to estimate additional parameters as is the case for the error splitting based combinations. As such this is the method we consider for dealing with lost data due to faults. To find new combination coefficients we need only solve the GCP problem for the set of multi-indices $I \backslash J$.

As discussed in Section 4.4.1, the GCP is difficult to solve in its most general form. Whilst it can be solved rather quickly if the poset $(I, \leq)$ is closed under $\wedge$ (i.e. a lower semi-lattice), this is no longer any help in the FTCT since the random nature of faults means we cannot guarantee that $(I \backslash J, \leq)$ is always a lower semi-lattice. The only way we could ensure this is to restrict which elements of I can be in J. A simple way to achieve this is to recompute missing $u_{\underline{i}}$ if $(I \backslash \{\underline{i}\}, \leq)$ is not a lower semi-lattice. In particular this is achieved if all $u_{\underline{i}}$ with $\underline{i} \notin \max I$ are recomputed. Since elements in $\max I$ correspond to the solutions on the largest of the grids, we are avoiding the recomputation of the solutions which take the longest to compute. This means that any delays caused by recomputation are less likely to occur and are much shorter compared to the 'local' checkpointing algorithm (if they do occur). Additionally, this also means only the largest of the hierarchical spaces are ever omitted as a result of a failure. As these contribute the least to the solution we expect the resulting error to be relatively close to that of $u_{\underline{i}}$, i.e. the computed solution in the event that no faults occur. Finally, since $(I\backslash J, \leq)$ is then a lower semi-lattice, the resulting GCP for $I\backslash J$ has a unique maximal solution which is easily computed.

We now illustrate this approach as it is applied to the classical combination technique. We define $I_n = \{\underline{i} \in \mathbb{N}^d : |\underline{i}| \leq n\}$. It was shown in [67] that the proportion of additional unknowns in computing the solutions $u_{\underline{i}}$ for all $\underline{i} \in I_n$ compared to $n - d < |\underline{i}| \leq n$ is at most $\frac{1}{2^d-1}$. If no faults occur then the combination is exactly the classical combination technique with $c_{\underline{i}} = (-1)^{n-|\underline{i}|}\binom{d-1}{n-|\underline{i}|}$ if $n - d < |\underline{i}| \leq n$ and $c_{\underline{i}} = 0$ otherwise. If faults do occur then we recompute any $u_{\underline{i}}$ with $|\underline{i}| < n$ that were not successfully computed. If no faults occurred for any $u_{\underline{i}}$ with $|\underline{i}| = n$ then we can again proceed with the classical combination. If faults affect any $u_{\underline{i}}$ with $|\underline{i}| = n$ then we add such $\underline{i}$ to the set J and then solve the GCP for $I_n\backslash J$. The solution is trivially obtained with hierarchical coefficients $\omega_{\underline{i}} = 1$ for all $\underline{i} \in I_n\backslash J$.

The largest solutions (in terms of unknowns) which may have to be recomputed are those with $|\underline{i}| = n - 1$ which would be expected to take at most half the time of those solutions with $|\underline{i}| = n$. Since they take less time to compute they are also less likely to be lost due to failure. Additionally, there are $\binom{n-1+d-1}{d-1}$ solutions with $|\underline{i}| = n - 1$ which is less than the $\binom{n+d-1}{d-1}$ with $|\underline{i}| = n$. As a result of these observations, we would expect to see far less disruptions caused by recomputation when using this approach compared to a lossless approach where all failed solutions are recomputed.

The worst case scenario with this approach is that all $u_{\underline{i}}$ with $|\underline{i}| = n$ are not successfully computed due to faults. In this case the resulting combination is simply a classical combination of level $n - 1$. This only requires the solutions $u_{\underline{i}}$ with $n - d \leq |\underline{i}| \leq n - 1$. Likewise, all solutions to the GCP in this approach result in zero coefficients for all $c_{\underline{i}}$ with $|\underline{i}| < n - d$. We can therefore reduce the overhead of the FTCT by only computing the solutions $u_{\underline{i}}$ for $n - d \leq |\underline{i}| \leq n$ (instead of all $u_{\underline{i}}$ with $|\underline{i}| \leq n$). It is known that the proportion of additional unknowns compared to the classical combination technique in this case is at most $\frac{1}{2(2^d-1)}$ [67]. This asymptotic estimate is perhaps misleading as for $d > 2$ and practical levels $n \leq 20$ this is an overestimate. It also neglects the fact that a load balancing for $u_{\underline{i}}$ with $n - d < |\underline{i}| \leq n$ will generally be imperfect and that the computation of the additional $u_{\underline{i}}$ with $|\underline{i}| = n - d$ will often fill the gaps in the load balancing without extending the total computation time. Further still, the algorithm could be adjusted to only compute the necessary $u_{\underline{i}}$ in the event of failures.

The solutions $u_{\underline{i}}$ with $|\underline{i}| = n-1$ have approximately half the unknowns of the largest $u_{\underline{i}}$ and the recomputation of these may still be disruptive and undesirable. We could therefore consider recomputing only solutions with $|\underline{i}| \leq n-2$. By doing this the recomputations are even more manageable having at most one quarter the unknowns of the largest $u_{\underline{i}}$. The worst case here is that all solutions with $|\underline{i}| \geq n-1$ fail and we end up with a classical combination of level $n-2$. Again it turns out one does not require the entire downset I_n, in this case the (modified) FTCT requires solutions $u_{\underline{i}}$ with $n - d - 1 \leq |\underline{i}| \leq n$. Using arguments similar to those in [67] it is easily shown that the overhead in this case is at most $\frac{3}{4(2^d-1)}$ (although again one typically has far less redundancy in practice). The trade-off now is that the update of coefficients takes a little more work. We are back in the situation where we cannot guarantee that $(I_n \backslash J, \leq)$ is a lower semi-lattice.

To solve the GCP in this case we start with all $\omega_{\underline{i}}$ equal to 1. If failures affected any $u_{\underline{i}}$ with $|\underline{i}| = n$ we set the corresponding constraints $c_{\underline{i}} = \omega_{\underline{i}} = 0$. For failures occurring on $u_{\underline{i}}$ with $|\underline{i}| = n - 1$ we have the constraints $\omega_{\underline{i}} - \sum_{k=1}^{d} \omega_{\underline{i}+\underline{e}^k} = 0$ (with $\underline{e}^k$ being the multi-index with $e^k_l = \delta_{k,l}$). We note that (since the $\omega_{\underline{i}}$ are binary variables) this can only be satisfied if at most one of the $\omega_{\underline{i}+\underline{e}^j}$ is equal to 1. Further, if $\sum_{k=1}^{d} \omega_{\underline{i}+\underline{e}^k} = 0$ we must also have $\omega_{\underline{i}} = 0$. This gives us a total of $d+1$ feasible solutions to check for each such constraint. Given g failures on solutions with $|\underline{i}| = n - 1$ we have at most $(d + 1)^g$ feasible solutions to the GCP to check. This can be kept manageable if solutions are combined frequently enough that the number of failures g that are likely occur in between is small. One solves the GCP by computing the objective function (4.16) for each of the feasible solutions identified and selecting one which maximises this. Where some of the failures on the second layer are sufficiently far apart on the lattice, it is possible to significantly reduce the number of cases to check as constraints can be optimised independently.

We could continue and describe an algorithm for only recomputing the fourth layer and below, however the coefficient updates here begin to become much more complex (both to describe and to compute). Our experience indicates that the recomputation of the third layer and below is a good trade-off between the need to recompute and the complexity of updating the coefficients. The numerical results in Section 5.2.4 are obtained using this approach.

5.2.1 Implementation of the FTCT

In Algorithm 3 we describe the parallel FTCT algorithm. Note that the pseudocode focuses on fault tolerance during the time consuming evolution of the $u_{\underline{i}}$ and would require some additional modifications to be fault tolerant through the combination stages. Here we elaborate on some details.

- On line 2 the partition is done in such a way that the workload on each process pool is balanced, that is given a work function $W : \mathbb{N}^d \to \mathbb{R}_+$ for which $W(\underline{i})$ is the amount of time required to compute the coarse approximation $u_{\underline{i}}$ then the I_p are chosen such that $\sum_{\underline{i} \in I_p} W(\underline{i}) \approx \frac{1}{P} \sum_{\underline{i} \in I} W(\underline{i})$ for all $p = 1, \dots, P$.

- When a fault occurs we assume that only the affected process pool need to be restarted (with all other process pools continuing uninterrupted). The restarted process then loads data from checkpoints and uses the values of the states $S_{\underline{i}}$ and n to continue execution from the correct location.

- The setting of $S_{\underline{i}} = -1$ on line 12 is meant to flag the $u_{\underline{i}}$ which we will not recompute in the event of a failure. In particular, only the smaller of the component solutions are typically recomputed upon failure.

- On line 17 the process pools communicate which $u_{\underline{i}}$ have been successfully computed. This allows all processes to compute combination coefficients which are consistent and only utilise those component solutions which are available (via the GCP).

- The combination of hierarchical surpluses in line 20 is first done on the local process over the $u_{\underline{i}}$ with $\underline{i} \in I_p$, the results of this are then combined globally. For example, implemented using the message passing interface (MPI) we use MPI_ALLREDUCE to sum the partial combination of the surpluses (premultiplied by the appropriate coefficients) and distribute the results to all processes.

Algorithm 3 Parallel fault tolerant combination technique algorithm. This describes the procedure on each process (or work group in a process pool). As with the local checkpointing algorithm we assume that when a process fails then only the failed process needs to be restarted (all others continue as usual). The restarted process is able to continue where it left off using the counter n and the states $S_{\underline{i}}$ possibly skipping the evolution of a $u_{\underline{i}}$ for which a failure occurred.

```
 1: if no checkpoint exists then
 2:    compute partition of $I = I_1 \cup \cdots \cup I_P$ amongst $P$ process pools
 3:    set $n = 0$ and partition $I_p$ associated with this process (with rank $p$)
 4:    initialise $u_{\underline{i}}$ and set state $S_{\underline{i}} = n$ for all $\underline{i} \in I_p$
 5:    checkpoint $n$, $I_p$ and the pairs $u_{\underline{i}}, S_{\underline{i}}$ for all $\underline{i} \in I_p$
 6: else
 7:    from checkpoint load $n$, the partition $I_p$, and the pairs $u_{\underline{i}}, S_{\underline{i}}$ for all $\underline{i} \in I_p$
 8: end if
 9: while $n < N$ do
10:    for each $\underline{i} \in I_p$ with $S_{\underline{i}} = n$ do
11:       if $u_{\underline{i}}$ is to be discarded upon failure then
12:          set $S_{\underline{i}} = -1$ and write to checkpoint
13:       end if
14:       evolve $u_{\underline{i}}$ by some fixed number of iterations
15:       set $S_{\underline{i}} = n + 1$ and checkpoint the pair $u_{\underline{i}}, S_{\underline{i}}$
16:    end for
17:    broadcast and gather the $S_{\underline{i}}$ for all $\underline{i} \in I$ // global communication
18:    compute combination coefficients $c_{\underline{i}}$ by solving the GCP for $J = \{\underline{i} : S_{\underline{i}} =$
       $n + 1\}$
19:    hierarchise each $u_{\underline{i}}$ for all $\underline{i} \in I_p$ for which $S_{\underline{i}} = n + 1$
20:    combine hierarchical surpluses // global communication
21:    reconstruct nodal basis for $u_{\underline{i}}$ for all $\underline{i} \in I_p$
22:    set $S_{\underline{i}} = n + 1$ and checkpoint the pair $u_{\underline{i}}, S_{\underline{i}}$ for all $\underline{i} \in I_p$
23:    set $n = n + 1$ and write to checkpoint
24: end while
```

- The reconstruction of $u_{\underline{i}}$ on line 21 includes those $u_{\underline{i}}$ which had previously failed. That is the $u_{\underline{i}}$ which were lost are estimated with the combination of the remaining $u_{\underline{i}}$.

- The evolution and combination parts of the code are decoupled in such a way that several evolution steps can be done between checkpoints, that is the checkpoint times and combination times can be decoupled.

At the time of writing this thesis there was limited support available for restarting individual process pools without restarting the entire job. We think the most promising software at this time is the User Level Fault Mitigation ULFM [11, 10] proposal developed by the MPI forum's fault tolerance working group for which the third beta release was available[1] at the time of writing this thesis. Mohsin Ali has done extensive work in implementing a version of our FTCT algorithm that supports real fault recovery using ULFM [3, 4, 5] and we report some results from this work in Section 5.2.4. However, my own work involved the validation of the FTCT algorithm via fault simulation for advection problems in two and three spatial dimensions.

When simulating faults we assume that the probability of failure occurring during the combination phase is negligible and therefore we to not implement fault simulation for this portion of the code. We base this assumption on profiling of the code which indicates that the combination is typically less than 1% of the total computation time for large computations with relatively infrequent combinations. Fault simulation was implemented within the parallel FTCT algorithm as follows.

- On line 4 each process (or work group in a process pool) samples a time of failure t_p before initialising the $u_{\underline{i}}$. The sample is made by sampling a distribution for the time to failure and adding this to the current time. In our experiments we sample the Weibull distribution with some mean $\lambda > 0$ and shape $0 < \kappa \leq 1$ which is set at run-time.

- Before writing the checkpoint on line 15 we check if the current wall time has exceeded the sampled time of failure t_p for that process. If this is the case then a failure is deemed to have occurred during the computation of the last $u_{\underline{i}}$ and the data is discarded. We then pretend that the effected process pool has been instantly restarted (although a delay can be inserted to simulate the time required to do this). A new time to failure is then

[1]`http://fault-tolerance.org/ulfm/downloads/`

sampled on the effected process pool and the algorithm then proceeds to determine if the $u_{\underline{i}}$ should be recomputed.

Due to limited data at the current time we are unable to predict what recovery times one might expect for the replacement of a failed node in an application. Some measurements we made using ULFM in [5] but as ULFM is still in beta we feel these results are not representative of what could be expected in a first release. Also note that (simulated) failures are checked for at the completion of the computation of each $u_{\underline{i}}$. Since a failure is most likely to occur some time before the computation completes then time is wasted in the simulation from the sampled time of failure to the completion of the affected computation. In practice, the failure of a process causing the loss of a grid that is not to be recomputed would cause the process to finish earlier than expected (provided the restart of the process is less than the compute time which remained for the lost solution). Depending on the load balancing this may in turn cause the application to finish sooner than expected.

5.2.2 Expected error of the FTCT for interpolation

In this section, we bound the expected interpolation error for the FTCT as applied to the classical combination technique as described in Section 5.2. In particular we look at the case where all solutions with $|\underline{i}| < n$ are recomputed, and the case where all solutions with $|\underline{i}| < n - 1$ are recomputed as described in Section 5.2. Although we do not do so here, it should be clear how these results can be extended to truncated combination starting from the estimates in Section 4.1.

Given $u \in H^2_{0,\text{mix}}$ then for each $\underline{i} \in \mathbb{N}^d$ let $u_{\underline{i}}$ be the piecewise multi-linear interpolant of u on the grid $\Omega_{\underline{i}} = \{\underline{j}2^{-\underline{i}} : \underline{0} \leq \underline{j} \leq 2^{\underline{i}}\}$ (with $\underline{j}2^{-\underline{i}} = (j_1 2^{-i_1}, \dots, j_d 2^{-i_d})$ and $2^{\underline{i}} = (2^{i_1}, \dots, 2^{i_d})$). Define

$$\epsilon_n := \frac{1}{3} 3^{-d} 2^{-2n} \|D^{\underline{2}} u\|_2 \sum_{k=0}^{d-1} \binom{n}{k} \left(\frac{1}{3}\right)^{d-1-k} \tag{5.2}$$

then the classical combination

$$u_n^c := \sum_{k=0}^{d-1} (-1)^k \binom{d-1}{k} \sum_{|\underline{i}|=n \,\&\, \underline{i} \geq \underline{1}} u_{\underline{i}} \,.$$

satisfies $\|u - u_n^c\|_2 \leq \epsilon_n$, see Proposition (2.19). Note that we can restrict $\underline{i} \geq \underline{1}$ because $u_{\underline{i}} = 0$ for $\underline{i} \not\geq \underline{1}$ as u is zero on the boundary. Therefore, in this subsection, we define the set of multi-indices $I_n := \{\underline{i} \in \mathbb{N}^d : |\underline{i}| \leq n \,\&\, \underline{i} \geq \underline{1}\}$. Thus, given $\{u_{\underline{i}}\}_{\underline{i} \in I_n}$ one is able compute u_n^c. Further, we note that it is straightforward to extend the work of this section to index sets that would be more useful for computing truncated combinations using the results of Section 4.1. When faults prevent successful computation of some of the $u_{\underline{i}}$ we must find $u_{I'}^{\mathrm{gcp}}$ for some $I' \subset I_n$ containing only those $\underline{i}$ for which $u_{\underline{i}}$ was successfully computed.

Consider the independent Bernoulli random variables $\{U_{\underline{i}}\}_{\underline{i} \in I_n}$ for which

$$U_{\underline{i}}(\sigma) := \begin{cases} 0 & \text{if } \sigma \text{ is the event that } u_{\underline{i}} \text{ is computed successfully} \\ 1 & \text{otherwise.} \end{cases} \tag{5.3}$$

We assume that the only event preventing the successful computation of $u_{\underline{i}}$ is the failure of at least one process involved in the computation of $u_{\underline{i}}$. Additionally it is assumed that no process is involved in the computation of more than one $u_{\underline{i}}$ at any given time such that the $U_{\underline{i}}$ are independent. Supposing that failures on each hardware node are accurately modelled by an ordinary renewal process, then $\Pr(U_{\underline{i}} = 1)$ depends on the number of hardware nodes over which the computation of $u_{\underline{i}}$ is distributed. We assume that a failure on any node involved in the computation of $u_{\underline{i}}$ results in the failure of the computation itself. Let $N_{\underline{i}}$ be the number of nodes, $Y_1, \dots, Y_{N_{\underline{i}}}$ be random variables for the time to next failure on each of the nodes (that is the forward recurrence times, which are IID) and $t_{\underline{i}}$ be the computation time (wall time), then

$$\begin{aligned} \Pr(U_{\underline{i}} = 1) &= 1 - \Pr(Y_1 > t_{\underline{i}}, \dots, Y_{N_{\underline{i}}} > t_{\underline{i}}) \\ &= 1 - \prod_{k=1}^{N_{\underline{i}}} \Pr(Y_k > t_{\underline{i}}) \\ &= 1 - \Pr(Y_1 > t_{\underline{i}})^{N_{\underline{i}}} . \end{aligned}$$

If the time between failures on each hardware node is exponentially distributed with mean λ then the forward recurrence time Y_1 is also exponentially distributed with mean λ (for asymptotically large starting times, see Theorem 1.13). In this case the probability of at least one failure on the $N_{\underline{i}}$ nodes is exponentially distributed with mean $\lambda/N_{\underline{i}}$, in particular we have $\Pr(U_{\underline{i}} = 1) = 1 - e^{-t_{\underline{i}} N_{\underline{i}}/\lambda}$. Similarly if the time between failures on each node is Weibull distributed with scale λ and shape $0 < \kappa \leq 1$ (and thus mean $\lambda\Gamma(1 + 1/\kappa)$) then the cumulative distribution of the forward recurrence time Y_1 is given by

$$\Pr(Y_1 \leq s) = \frac{1}{\lambda} \int_0^s e^{(r/\lambda)^\kappa} \, dr \, ,$$

(again this is for asymptotically large starting times, see Theorem 1.13). From Proposition 1.22 we know that that $\Pr(Y_1 \leq s) \leq \Pr(X \leq s)$ (with X Weibull distributed with scale λ and shape κ). As a consequence it also follows that

$$1 - \Pr(Y_1 > t_{\underline{i}})^{N_{\underline{i}}} \leq 1 - \Pr(X_1 > t_{\underline{i}})^{N_{\underline{i}}} \,,$$

and therefore

$$\Pr(U_{\underline{i}} = 1) \leq 1 - e^{-N_{\underline{i}}(t_{\underline{i}}/\lambda)^{\kappa}} \,.$$

In particular we obtain an upper bound for $\Pr(U_{\underline{i}} = 1)$ by modelling the probability of at least one failure on the $N_{\underline{i}}$ nodes during the computation by a Weibull random variable having scale $\lambda/N_{\underline{i}}^{1/\kappa}$ and shape κ.

Given that we have shown that for computations distributed over several hardware nodes one may obtain an upper bound for $\Pr(U_{\underline{i}} = 1)$ by simply adjusting the scale λ (or equivalently the mean $\lambda\Gamma(1 + 1/\kappa)$) appropriately we simplify the calculations that follow by assuming that the computation of each $u_{\underline{i}}$ is performed within one hardware node. Further, as $u_{\underline{i}}$ with identical $|\underline{i}|$ have a similar number of unknowns it will be assumed that the time to compute such $u_{\underline{i}}$ is also similar. In particular we define $t_n := \max_{|\underline{i}|=n} t_{\underline{i}}$ such that $t_{\underline{i}} \leq t_{|\underline{i}|}$ for all $\underline{i}$ and therefore

$$\Pr(U_{\underline{i}} = 1) = \Pr(Y_1 \leq t_{\underline{i}}) \leq \Pr(Y_1 \leq t_{|\underline{i}|}) \leq \Pr(X \leq t_{|\underline{i}|}) = 1 - e^{-(t_{|\underline{i}|}/\lambda)^{\kappa}} \,.$$

Consider the situation in which we recompute any $u_{\underline{i}}$ which fail if $|\underline{i}| < n$. We define the random vector $U_{n,1} = (U_{\underline{i}})_{\underline{i} \in I_n \backslash I_{n-1}}$ which contains the random variables $U_{\underline{i}}$ which indicate the success or failure of the corresponding $u_{\underline{i}}$. As $I_n \backslash I_{n-1}$ has $\binom{n-1}{d-1}$ elements there are $2^{\binom{n-1}{d-1}}$ possible states of the random vector $U_{n,1}$, namely $U_{n,1} \in \{0,1\}^{\binom{n-1}{d-1}}$. More generally we have the following definition.

Definition 5.3. Given $s, n \in \mathbb{N}$ with $1 \leq s \leq n$ and then $U_{n,s}$ is defined as the random vector

$$U_{n,s} := (U_{\underline{i}})_{\underline{i} \in I_n \backslash I_{n-s}} \,,$$

with the $U_{\underline{i}}$ as defined in (5.3) (which are independent). Further we define the support of $U_{n,s}$ as

$$\operatorname{supp}(U_{n,s}) := \{\underline{i} \in I_n \backslash I_{n-s} : U_{\underline{i}} = 1\} \,.$$

If the $u_{\underline{i}}$ with $\underline{i} \in I_{n-s}$ are recomputed upon failure then the set of multi-indices corresponding to the successfully computed $u_{\underline{i}}$ is given by $I_n \backslash \operatorname{supp}(U_{n,s})$. Given $U_{n,s}$ then the solution of the fault tolerant combination technique with combination coefficients given by a solution to the GCP for the set $I_n \backslash \operatorname{supp}(U_{n,s})$ is denoted by $u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,s})}$, that is

$$u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,s})} := \sum_{\underline{i} \in I_n \backslash \operatorname{supp}(U_{n,s})} c_{\underline{i}} u_{\underline{i}} ,$$

where the $c_{\underline{i}}$ are given by a GCP solution for the set $I_n \backslash \operatorname{supp}(U_{n,s})$ and each $u_{\underline{i}}$ is the piecewise multilinear interpolant of u on the grid $\Omega_{\underline{i}}$. Note that $u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,s})}$ is effectively a function of the random variable $U_{n,s}$ and is therefore also a random variable. With this we can now give the first result which bounds the expectation of the interpolation error if only those $u_{\underline{i}}$ with $|\underline{i}| < n$ are recomputed if lost due to a fault.

Proposition 5.4. *Fix the dimension $d > 0$ and let $n \geq d$. Let $u \in H^2_{0,mix}([0,1]^d)$ and for finite $J \subset \mathbb{N}^d$ let $u^{gcp}_J := \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}}$ where the $u_{\underline{i}}$ are piecewise multi-linear interpolants of u on the grid $\Omega_{\underline{i}} = \{\underline{j} 2^{-\underline{i}} : \underline{0} \leq \underline{j} \leq 2^{\underline{i}}\}$ and the $c_{\underline{i}}$ are given by a solution of the GCP for the set J. For each $u_{\underline{i}}$ we define $t_{\underline{i}}$ to be the time required to compute $u_{\underline{i}}$ and set $t_n = \max_{|\underline{i}|=n} t_{\underline{i}}$. Let ϵ_n be as defined in (5.2). Let $U_{n,s}$ be the random vector defined in Definition 5.3 with $s = 1$ and $\Pr(U_{\underline{i}} = 1) \leq 1 - e^{-(t_{\underline{i}}/\lambda)^{\kappa}}$ for each $\underline{i}$ with $\lambda > 0$ and $0 < \kappa \leq 1$. Then*

$$\mathbb{E}\left[\|u - u^{gcp}_{I_n \backslash \operatorname{supp}(U_{n,1})}\|_2\right] \leq \epsilon_n \left(1 + 3\left(1 - e^{-(t_n/\lambda)^{\kappa}}\right)\right) . \tag{5.4}$$

Proof. Consider a sample $J = I_n \backslash \operatorname{supp}(U_{n,1})$. We observe that $I_{n-1} \subseteq J \subseteq I_n$ and J is a lower-semilattice (i.e. closed under $\wedge$). It follows that there is a unique solution to the GCP for the set J (in particular $\omega_{\underline{i}} = 1$ for all $\underline{i} \in J{\downarrow}$ and $\omega_{\underline{i}} = 0$ otherwise, see Section 4.4). Additionally, the resulting coefficients $c_{\underline{i}}$ which are non-zero satisfy $\underline{i} \in J$. Thus $u^{\mathrm{gcp}}_J = u_J := P_J u$ with P_J as described in Section 4.2. Further, we may apply Proposition 4.26 to obtain

$$\|u - u_J\|_2 \leq 3^{-d} \|D^{\underline{2}} u\|_2 \left(3^{-d} - \sum_{\underline{i} \in J} 2^{-2|\underline{i}|}\right) ,$$

which we may decompose into

$$\|u - u_J\|_2 \leq 3^{-d} \|D^{\underline{2}} u\|_2 \left(3^{-d} - \sum_{\underline{1} \leq \underline{i} \in I_n} 2^{-2|\underline{i}|}\right) + 3^{-d} \|D^{\underline{2}} u\|_2 \sum_{\underline{i} \in I_n \backslash J} 2^{-2|\underline{i}|} .$$

Note that the left term bounds $\|u - u^c_n\|_2$ which is in turn bounded by ϵ_n (Proposition 2.19). Also, as $I_n \backslash J = \operatorname{supp}(U_{n,1})$ and $\underline{i} \in \operatorname{supp}(U_{n,1})$ implies that $|\underline{i}| = n$

and $U_{\underline{i}} = 1$ then it follows that

$$\begin{aligned}\|u - u^{\text{gcp}}_{I_n \setminus \operatorname{supp}(U_{n,1})}\|_2 &\leq \epsilon_n + 3^{-d}\|D^{\underline{2}}u\|_2 \sum_{\underline{i} \in \operatorname{supp}(U_{n,1})} 2^{-2|\underline{i}|} \\ &= \epsilon_n + 3^{-d}\|D^{\underline{2}}u\|_2 \sum_{\underline{i} \in I_n \setminus I_{n-1}} U_{\underline{i}} 2^{-2n} \,.\end{aligned}$$

We now take the expectation of both sides. As $\mathrm{E}[U_{\underline{i}}] = \Pr(U_{\underline{i}} = 1)$ and expectation is linear and monotone one has

$$\begin{aligned}\mathbb{E}\left[\|u - u^{\text{gcp}}_{I_n \setminus \operatorname{supp}(U_{n,1})}\|_2\right] &\leq \epsilon_n + 3^{-d}\|D^{\underline{2}}u\|_2 \sum_{\underline{i} \in I_n \setminus I_{n-1}} 2^{-2n} \Pr(U_{\underline{i}} = 1) \\ &\leq \epsilon_n + 3^{-d}\|D^{\underline{2}}u\|_2 \sum_{\underline{i} \in I_n \setminus I_{n-1}} 2^{-2n} \left(1 - e^{-(t_{|\underline{i}|}/\lambda)^\kappa}\right) \\ &\leq \epsilon_n + 3^{-d}\|D^{\underline{2}}u\|_2 \binom{n-1}{d-1} 2^{-2n} \left(1 - e^{-(t_n/\lambda)^\kappa}\right) \,,\end{aligned}$$

Lastly we observe that

$$3^{-d}\|D^{\underline{2}}u\|_2 \binom{n-1}{d-1} 2^{-2n} \leq 3^{-d}\|D^{\underline{2}}u\|_2 \binom{n-1}{d-1} 2^{-2n} \sum_{k=0}^{d-1} \binom{n}{k} \left(\frac{1}{3}\right)^{d-1-k} = 3\epsilon_n$$

and therefore

$$\mathbb{E}\left[\|u - u^{\text{gcp}}_{I_n \setminus \operatorname{supp}(U_{n,1})}\|_2\right] \leq \epsilon_n + 3\epsilon_n \left(1 - e^{-(t_n/\lambda)^\kappa}\right) \,.$$

Collecting the common factor ϵ_n yields the desired result. □

Note that as $t_n/\lambda \to \infty$ we have $\mathbb{E}\left[\|u - u^{\text{gcp}}_{I_n \setminus \operatorname{supp}(U_{n,1})}\|_2\right] \leq 4\epsilon_n$. However, the worst case scenario is when $\operatorname{supp}(U_{n,1}) = I_n \setminus I_{n-1}$ which results in a classical combination of level $n-1$ which has the error bound

$$\begin{aligned}\|u - u^c_{n-1}\|_2 \leq \epsilon_{n-1} &= \frac{1}{3} \cdot 3^{-d} 2^{-2(n-1)} \|D^{\underline{2}}u\|_2 \sum_{k=0}^{d-1} \binom{n-1+d}{k} \left(\frac{1}{3}\right)^{d-1-k} \\ &\leq \frac{4}{3} \cdot 3^{-d} 2^{-2n} \|D^{\underline{2}}u\|_2 \sum_{k=0}^{d-1} \binom{n+d}{k} \left(\frac{1}{3}\right)^{d-1-k} \\ &= 4 \cdot \epsilon_n \,.\end{aligned}$$

This is consistent with the upper bound (5.4).

Notice that we sacrificed some tightness in the proof of Proposition 5.4 in order to express the bound as a multiple of ϵ_n. The reason for doing this is is that one obtains some indication for what the expected relative increase in error is when faults are of concern. In particular, if the bound $\|u - u_n^c\|_2 \leq \epsilon_n$ was tight, one might expect

$$\mathbb{E}\left[\|u - u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,1})}\|_2\right] \lessapprox \|u - u_n^c\|_2 \left(1 + 3\left(1 - e^{-(t_n/\lambda)^\kappa}\right)\right) ,$$

for which we see the relative increase is $3\left(1 - e^{-(t_n/\lambda)^\kappa}\right)$.

Now consider the situation in which we recompute any $u_{\underline{i}}$ which fail if $|\underline{i}| < n-1$. In this scenario the random vector $U_{n,2}$ (see Definition 5.3) tells us the state of those $u_{\underline{i}}$ with $\underline{i} \in I_n \backslash I_{n-2}$. As $I_n \backslash I_{n-2}$ has $\binom{n-1}{d-1} + \binom{n-2}{d-1}$ elements there are $2^{\binom{n-1}{d-1}+\binom{n-2}{d-1}}$ possible outcomes. The set of multi-indices corresponding to the successfully computed $u_{\underline{i}}$ is $I_n \backslash \mathrm{supp}(U_{n,2})$ (which contains I_{n-2}) and $u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,2})}$ is the corresponding random variable denoting the output of the fault tolerant combination technique. We now prove an error bound for this scenario analogous to Proposition 5.4.

Proposition 5.5. *Fix the dimension $d > 0$ and $n \geq d$. Let $u \in H^2_{0,mix}([0,1]^d)$ and for each finite $J \subset \mathbb{N}^d$ let u_J^{gcp} be as defined in Proposition 5.4. We again define $t_{\underline{i}}$ as the time required to compute $u_{\underline{i}}$ and set $t_n = \max_{|\underline{i}|=n} t_{\underline{i}}$ and similarly for t_{n-1}. Let ϵ_n be as defined in* (5.2). *Let $U_{n,s}$ be the random variable defined in Definition 5.3, $s = 2$ and $\Pr(U_{\underline{i}} = 1) \leq 1 - e^{-(t_{\underline{i}}/\lambda)^\kappa}$ with $\lambda > 0$ and $0 < \kappa \leq 1$. Then*

$$\mathbb{E}\left[\|u - u^{gcp}_{I_n \setminus \mathrm{supp}(U_{n,2})}\|_2\right] \\ \leq \epsilon_n \cdot \min\left\{16, 1 + 3\left(d + 5 - e^{-\left(\frac{t_n}{\lambda}\right)^\kappa} - (d+4)e^{-\left(\frac{t_{n-1}}{\lambda}\right)^\kappa}\right)\right\} .$$

Proof. Unlike the situation in Proposition 5.4, given a sample $J = I_n \backslash \mathrm{supp}(U_{n,2})$ the solution to the GCP may not be unique. However, we observe that there exists a largest downset J' (with respect to $\mathbb{N}_+^d$ with $\mathbb{N}_+ = \{1, 2, 3, \dots\}$) such that $I_{n-2} \subseteq J' \subseteq J$. The gcp has a unique solution for the set J' which corresponds to the combination $u_{J'} := P_{J'\downarrow} u$. As $u_{J'}$ is a candidate solution to the GCP for the set J any actual solution must satisfy $\|u - u_J^{\mathrm{gcp}}\|_2 \leq \|u - u_{J'}\|_2$. Further, J' is obtained by taking I_n and removing all $\underline{i}$ such that $\underline{i} \geq \underline{j}$ for some $\underline{j} \in I_n \backslash J = \mathrm{supp}(U_{n,2})$. In particular, given $\underline{j} \in \mathrm{supp}(U_{n,2})$ with $|\underline{j}| = n$ we need only remove $\underline{j}$. On the other hand, for $|\underline{j}| = n-1$ we remove $\underline{j}$ and $\underline{j} + \underline{e}^m$ for $m = 1, \dots, d$ (where $e_k^m = \delta_{m,k}$). By applying the result of Proposition 4.26 and decomposing in a manner similar to Proposition 5.4 we observe

$$\begin{aligned}
\|u - u_{J'}\|_2 &\leq 3^{-d}\|D^{\underline{2}}u\|_2 \left(3^{-d} - \sum_{\underline{i}\in J'} 2^{-2|\underline{i}|}\right) \\
&\leq 3^{-d}\|D^{\underline{2}}u\|_2 \left(3^{-d} - \sum_{\underline{i}\in I_n} 2^{-2|\underline{i}|}\right) + 3^{-d}\|D^{\underline{2}}u\|_2 \sum_{\underline{i}\in I_n\backslash(J'\cup I_{n-1})} 2^{-2|\underline{i}|} \\
&\qquad + 3^{-d}\|D^{\underline{2}}u\|_2 \sum_{\underline{i}\in I_{n-1}\backslash(J'\cap I_{n-1})} 2^{-2|\underline{i}|} .
\end{aligned}$$

As before, the first term is bounded above ϵ_n. The third term consists of only those $|\underline{i}| = n-1$ for which $u_{\underline{i}}$ was not successfully computed, that is $\underline{i} \in \operatorname{supp}(U_{n,2})$ with $|\underline{i}| = n-2$. For the second term we have $\underline{i} \in I_n\backslash(J' \cup I_{n-1})$ if $|\underline{i}| = n$ and at least one of the following is true

- $u_{\underline{i}}$ was not successfully computed (that is $\underline{i} \in \operatorname{supp}(U_{n,2})$),
- $u_{\underline{j}}$ was not successfully computed for some $|\underline{j}| = n-1$ with $\underline{i} \geq \underline{j}$ (that is $\underline{i} - \underline{e}^m \in \operatorname{supp}(U_{n,2})$ for some $m = 1, \dots, k$).

Thus, as $\underline{i} \in \operatorname{supp}(U_{n,2})$ if and only if $U_{\underline{i}} = 1$, one has

$$\|u - u^{\text{gcp}}_{J'}\|_2 \leq \epsilon_n + 3^{-d}\|D^{\underline{2}}u\|_2 \left(\sum_{\underline{i}\in I_n\backslash I_{n-1}} 2^{-2|\underline{i}|} \left(U_{\underline{i}} + \sum_{m=1}^{d} U_{\underline{i}-\underline{e}^m}\right) + \sum_{\underline{i}\in I_{n-1}\backslash I_{n-2}} 2^{-2|\underline{i}|} U_{\underline{i}}\right) .$$

We observe that $\underline{i} \in I_n\backslash I_{n-1}$ implies $|\underline{i}| = n$ and similarly $\underline{i} \in I_{n-1}\backslash I_{n-2}$ implies $|\underline{i}| = n-1$. Additionally, as $\|u - u^{\text{gcp}}_J\|_2 \leq \|u - u_{J'}\|_2$ and $J = I_n \backslash \operatorname{supp}(U_{n,2})$ one obtains

$$\|u - u^{\text{gcp}}_{I_n\backslash \operatorname{supp}(U_{n,2})}\|_2 \leq \epsilon_n + 3^{-d}\|D^{\underline{2}}u\|_2 \left(\sum_{\underline{i}\in I_n\backslash I_{n-1}} 2^{-2n} \left(U_{\underline{i}} + \sum_{m=1}^{d} U_{\underline{i}-\underline{e}^m}\right) + \sum_{\underline{i}\in I_{n-1}\backslash I_{n-2}} 2^{-2(n-1)} U_{\underline{i}}\right) .$$

Taking the expectation of both sides (and using the fact expectation is linear and monotone) we have

$$\mathrm{E}\left[\|u-u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})}\|_2\right] \leq \epsilon_n + 3^{-d}\|D^{\underline{2}}u\|_2 2^{-2n}\times \\ \left(\sum_{\underline{i}\in I_n\setminus I_{n-1}}\left(\mathrm{E}[U_{\underline{i}}]+\sum_{m=1}^{d}\mathrm{E}[U_{\underline{i}-\underline{e}^m}]\right)+4\sum_{\underline{i}\in I_{n-1}\setminus I_{n-2}}\mathrm{E}[U_{\underline{i}}]\right).$$

Now as $\mathrm{E}[U_{\underline{i}}]=\Pr(U_{\underline{i}}=1)\leq 1-e^{-(t_{|\underline{i}|}/\lambda)^\kappa}$ we have

$$\begin{aligned}\mathrm{E}&\left[\|u-u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})}\|_2\right]\\ &\leq \epsilon_n+3^{-d}\|D^{\underline{2}}u\|_2 2^{-2n}\left(\sum_{\underline{i}\in I_n\setminus I_{n-1}}\left(1-e^{-(t_n/\lambda)^\kappa}+\sum_{m=1}^{d}1-e^{-(t_{n-1}/\lambda)^\kappa}\right)\right.\\ &\qquad\left.+4\sum_{\underline{i}\in I_{n-1}\setminus I_{n-2}}1-e^{-(t_{n-1}/\lambda)^\kappa}\right)\\ &=\epsilon_n+3^{-d}\|D^{\underline{2}}u\|_2 2^{-2n}\left(\binom{n-1}{d-1}\left((1-e^{-(t_n/\lambda)^\kappa})+d(1-e^{-(t_{n-1}/\lambda)^\kappa})\right)\right.\\ &\qquad\left.+4\binom{n-2}{d-1}(1-e^{-(t_{n-1}/\lambda)^\kappa})\right)\\ &\leq\epsilon_n\left(1+3(1-e^{-(t_n/\lambda)^\kappa})+3\left(d+4\frac{n-d}{n-1}\right)(1-e^{-(t_{n-1}/\lambda)^\kappa})\right),\end{aligned}$$

where the last line uses the inequality $3^{-d}\|D^{\underline{2}}u\|_2 2^{-2n}\binom{n-1}{d-1}\leq 3\epsilon_n$ derived in Proposition 5.4. Noting that $\frac{n-d}{n-1}\leq 1$ and re-arranging we obtain

$$\mathrm{E}\left[\|u-u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})}\|_2\right]\leq\epsilon_n\left(1+3\left(d+5-e^{-\left(\frac{t_n}{\lambda}\right)^\kappa}-(d+4)e^{-\left(\frac{t_{n-1}}{\lambda}\right)^\kappa}\right)\right).$$

Now the expected error can be no more than the worse case $\mathrm{supp}(U_{n,2})=I_n\setminus I_{n-2}$ corresponding to the classical combination of level $n-2$ for which it is straightforward to show $\|u-u^c_{n-2}\|_2\leq 16\epsilon_n$. Taking the minimum of the two bounds yields the desired result. □

This result is actually an over-estimate of the error bound. The main reason for this is that our bound for the approximation $u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})}$ always excludes the hierarchical surplus $u^h_{\underline{j}}$ if $|\underline{j}|=n-1$ and $u_{\underline{j}}$ is not successfully computed. In practice the actual GCP solution will only exclude the surplus $u^h_{\underline{j}}$ if there are several $u_{\underline{i}}$ which have also failed for $\underline{i}$ in a neighbourhood of $\underline{j}$. For example if there has only been one failure resulting in the loss of $u_{\underline{j}}$ for $|\underline{j}|=n-1$ then the GCP solution is able to retain the surplus $u^h_{\underline{j}}$ by instead removing $d-1$ surpluses

$u_{\underline{i}}^h$ with $|\underline{i}| = n$, that is the factor $d + 4$ can quite often be replaced with $d - 1$. A better bound could be computed by enumerating all possible outcomes and bounding the GCP solution for each. However, as the total number of outcomes is $2^{\binom{n-1}{d-1}+\binom{n-2}{d-1}}$ this becomes cumbersome even for modest dimensions and levels.

To illustrate how this result may be used in practice, suppose we compute a level $n = 12$ interpolation in $d = 3$ dimensions on a machine whose mean time to failure can be modelled by the Weibull distribution with scale $\lambda = 1000$ and shape parameter $\kappa = 0.7$ (thus the mean time between failures is ≈ 1266 seconds). Further, suppose $u_{\underline{i}}$ with $|\underline{i}| > 10$ are not recomputed if lost as a result of a fault and that t_{12} is estimated to be 1.0 seconds and t_{11} is at most 0.5 seconds. The expected error for our computation is bounded above by 1.126 times the error bound if no faults were to occur, i.e. a relative increase of 12.6%. This may seem somewhat large initially but to put this into perspective suppose that $\|D^{\underline{2}}u\|_2 = 1$ then $\epsilon_n \approx 5.16\times10^{-8}$ and an increase of 12.6% leads to an upper bound of 5.81×10^{-8} for the expected error which for most practical purposes is not likely to make a perceivable difference in the solution. Even the worst case scenario which has error bounded above by 8.25×10^{-7} is quite small and certainly much better than no solution at all.

Proposition 5.5 gives some insight into how one may construct a bound for more general $U_{n,s}$ with $s = 3, ..., n - d$. Despite not being able to explicitly write down all of the GCP solutions in advance we can always find one candidate solution for which any GCP solution must improve upon. However, as with Proposition 5.5 this tends to lead to an over estimate. The over estimate is particularly significant for $s > d$ as here a single failure on a grid $\underline{i}$ with $|\underline{i}| \leq n - d$ gives quite a poor error bound despite the fact that the GCP solution is just the combination u_{I_n}. As such we have not included a bound for more general $U_{n,s}$ here.

So far we have only studied the expectation of interpolants computed with the fault-tolerant combination technique. Of course it would be nice to be able to say something about the variance (or standard deviation) as well. However it is very difficult to obtain a reasonable bound for this as we only have an upper bound for $\mathrm{E}[\|u^{\mathrm{gcp}}_{U_n\setminus \mathrm{supp}(U_{n,2})} - u\|_2]$. For $s = 1$ and $s = 2$ we can bound $\|u^{\mathrm{gcp}}_{U_n\setminus \mathrm{supp}(U_{n,2})} - u\|_2$ from above by $4\epsilon_n$ and $16\epsilon_n$ respectively and thus via Popoviciu's inequality their variances are bounded above by $4\epsilon_n^2$ and $64\epsilon_n^2$ respectively (noting that the lower bound is zero). Studying the variance (or standard deviation) is useful for understanding the spread of results from the mean. However, in our computations it would be just as, if not more, useful to bound the spread of results from that obtained in the absence of failures. In particular we study $\|u_{I_n} - u^{\mathrm{gcp}}_{I_n\setminus \mathrm{supp}(U_{n,s})}\|_2^2$

for the case $s = 1$.

Proposition 5.6. *Consider the same assumptions as in Proposition 5.4. Then*

$$\mathrm{E}\left[\|u_{I_n} - u^{gcp}_{I_n\setminus\mathrm{supp}(U_{n,1})}\|_2^2\right] \leq 9\epsilon_n^2\left(1-e^{-(t_n/\lambda)^\kappa}\right)\left(1-e^{-(t_n/\lambda)^\kappa}+\frac{e^{-(t_n/\lambda)^\kappa}}{\binom{n-1}{d-1}}\right).$$

Proof. Observe that

$$u_{I_n} - u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})} = \sum_{\underline{i}\in I_n\setminus I_{n-1}} U_{\underline{i}} u^h_{\underline{i}}$$

where each $u^h_{\underline{i}}$ is the $\underline{i}$th hierarchical surplus of u. It follows that

$$\|u_{I_n} - u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}\|_2^2 \leq \sum_{\underline{i},\underline{j}\in I_n\setminus I_{n-1}} U_{\underline{i}}U_{\underline{j}}\|u^h_i\|_2\|u^h_j\|_2 .$$

As expectation is linear and monotone it follows that

$$\mathrm{E}\left[\|u_{I_n} - u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}\|_2^2\right] \leq \sum_{\underline{i},\underline{j}\in I_n\setminus I_{n-1}} \mathrm{E}[U_{\underline{i}}U_{\underline{j}}]\|u^h_i\|_2\|u^h_j\|_2 .$$

Observe that $\mathrm{E}[U_{\underline{i}}U_{\underline{j}}] = \Pr(U_{\underline{i}}U_{\underline{j}} = 1)$. Further, for $\underline{i} \neq \underline{j}$ one has

$$\Pr(U_{\underline{i}}U_{\underline{j}} = 1) = \Pr(U_{\underline{i}} = 1)\Pr(U_{\underline{j}} = 1) \leq (1-e^{-(t_n/\lambda)^\kappa})^2$$

whilst for $\underline{i} = \underline{j}$ one has $\Pr(U_{\underline{i}}U_{\underline{j}} = 1) = \Pr(U_{\underline{i}} = 1) \leq 1-e^{-(t_n/\lambda)^\kappa}$. Further, as $|\underline{i}| = n$ for $\underline{i}\in I_n\setminus I_{n-1}$ then

$$\|u^h_{\underline{i}}\|_2 \leq 3^{-d}\|D^{\underline{2}}u\|_2 2^{-2n} \leq \frac{3\epsilon_n}{\binom{n-1}{d-1}} .$$

Lastly, as $|I_n\setminus I_{n-1}| = \binom{n-1}{d-1}$ we obtain

$$\begin{aligned}\mathrm{E}\left[\|u_{I_n} - u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}\|_2^2\right] \leq{}& \binom{n-1}{d-1}\left(\binom{n-1}{d-1}-1\right)\frac{9\epsilon_n^2}{\binom{n-1}{d-1}^2}(1-e^{-(t_n/\lambda)^\kappa})^2\\ &+\binom{n-1}{d-1}\frac{9\epsilon_n^2}{\binom{n-1}{d-1}^2}(1-e^{-(t_n/\lambda)^\kappa}) .\end{aligned}$$

Collecting the common terms and re-arranging yields the desired result. □

Remark 5.7. Note that for interpolation the GCP solutions will sometimes throw away some information that has been computed, that is some $u_{\underline{i}}$ which were successfully computed may not be used due to a failure. In practice one may add contributions from such $u_{\underline{i}}$ via the hierarchical decomposition. In this case a hierachical surplus $u^h_{\underline{i}}$ does not contribute only if $u_{\underline{j}}$ is not successfully computed for all $\underline{j} \geq \underline{i}$. Thus the probability that $u^h_{\underline{i}}$ contributes to the error is $\prod_{\underline{j}\geq\underline{i}} \Pr(U_{\underline{i}} = 1)$. Whilst one could provide an error bounds based on these probabilities interpolation we do not do so here. The time required to compute interpolants will typically be fast enough that the probability of failure is negligible and even in the event of failure it is not likely to be inconvenient to recompute. Our results here are meant to give an indication of the expected error for more complex and time consuming computations for which the approximations $u_{\underline{i}}$ are close to the piecewise multi-linear interpolant of the true solution u.

5.2.3 Results for point-wise error splitting

Whilst knowing we get reasonable results for interpolation is a good start, what we are really interested in is whether we can expect to obtain good results for approximate solutions of partial differential equation (PDE's). In particular we are interested in the advection equation, for which we showed in Section 3.3 that certain finite difference methods lead to an approximation which satisfies the classical error splitting model (see (3.24) and (2.22)). Thus our goal in this section is to obtain error bounds for the fault tolerant combination technique when coarse solutions satisfy the error splitting model.

We start with an analysis of the two dimensional case for which we assume approximations $u_{i,j}$ of u satisfy

$$u_{i,j}(\boldsymbol{x}) - u(\boldsymbol{x}) = \epsilon_x(\boldsymbol{x}, h_i)h_i^p + \epsilon_y(\boldsymbol{x}, h_j)h_j^p + \epsilon_{xy}(\boldsymbol{x}, h_i, h_j)h_i^p h_j^p \,, \tag{5.5}$$

with $h_i := 2^{-i}$, $p \geq 1$ and $|\epsilon_x|, |\epsilon_y|, |\epsilon_{xy}| \leq K$ for some $K > 0$ for all $\boldsymbol{x}, h_i, h_j$. We typically drop the $\boldsymbol{x}$ argument for ease of notation. Unlike the previous section in which we considered $u \in H^2_{0,\text{mix}}$ we do not make this assumption here. In particular we allow functions which are non-zero on the boundary. Therefore in this section we consider index sets $I_n = \{\underline{i} \in \mathbb{N}^d : |\underline{i}| \leq n\}$, i.e. unlike the previous section we now allow the components of $\underline{i}$ to be zero. For these sets we define $U_{n,s}$ analogous to that in the previous section, that is $U_{n,s} = (U_{\underline{i}})_{\underline{i}\in I_n\setminus I_{n-s}}$, but with the I_n including $\underline{i}$ having zero components. This random vector can again be mapped to the result of the fault tolerant combination technique via

$$u^{\text{gcp}}_{I_n\setminus\text{supp}(U_{n,s})} := \sum_{\underline{i}\in I_n\setminus\text{supp}(U_{n,s})} c_{\underline{i}} u_{\underline{i}} \,,$$

where the $c_{\underline{i}}$ are determined by a GCP solution for the set $I_n \backslash \operatorname{supp}(U_{n,s})$ and the $u_{\underline{i}}$ are approximations of u satisfying a pointwise error splitting, e.g. (5.5) in 2 dimensions. As with the interpolation estimates, it should be clear how the following results can be extended to truncated combinations (by translating the index set I_n and building on the estimates given in Section 4.1) but for brevity we do not develop this here.

Consider the case $s = 1$ where $u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,1})}$ is an approximation obtained from the fault tolerant combination technique when we do not recompute $u_{\underline{i}}$ which fail if $|\underline{i}| = n$ (as described in Section 5.2.2). For these combinations with $d = 2$ we have the following lemma.

Lemma 5.8. *Let $d = 2$ and $I_{n-1} \subset J \subseteq I_n$ then J is a downset and the combination u_J is given by*

$$u_J = \sum_{|\underline{i}|=n} u_{\underline{i}} - \sum_{|\underline{i}|=n-1} u_{\underline{i}} - \sum_{\substack{|\underline{i}|=n \\ \underline{i} \geq \underline{1}}} \chi_{I_n \backslash J}(\underline{i})(u_{i_1,i_2} - u_{i_1-1,i_2} - u_{i_1,i_2-1} + u_{i_1-1,i_2-1})$$
$$- \chi_{I_n \backslash J}((n,0))(u_{n,0} - u_{n-1,0}) - \chi_{I_n \backslash J}((0,n))(u_{0,n} - u_{0,n-1}) .$$

Proof. A consequence of Proposition 4.19 is that we may write the combination u_J as

$$u_J = \sum_{\underline{i} \in J} u_{i_1,i_2} - u_{i_1-1,i_2} - u_{i_1,i_2-1} + u_{i_1-1,i_2-1}$$

where $u_{i_1,i_2} := 0$ if $i_1 < 0$ and/or $i_2 < 0$. Similar applies for u_{I_n} such that one has

$$u_J - u_{I_n} = - \sum_{\underline{i} \in I_n \backslash J} u_{i_1,i_2} - u_{i_1-1,i_2} - u_{i_1,i_2-1} + u_{i_1-1,i_2-1} .$$

Now consider those $\underline{i} \in I_n \backslash J$ for which at least one of i_1, i_2 are zero. There are only two such $\underline{i}$, namely $(n, 0)$ and $(0, n)$ and for these two values we observe that the neighbours satisfy $u_{n,-1} = u_{n-1,-1} = 0$ and $u_{-1,n} = u_{-1,n-1} = 0$ respectively. Thus we obtain

$$u_{I_n} - u_J = \sum_{\substack{\underline{i} \in I_n \backslash J \\ \underline{i} \geq \underline{1}}} (u_{i_1,i_2} - u_{i_1-1,i_2} - u_{i_1,i_2-1} + u_{i_1-1,i_2-1})$$
$$+ \chi_{I_n \backslash J}((n,0))(u_{n,0} - u_{n-1,0}) + \chi_{I_n \backslash J}((0,n))(u_{0,n} - u_{0,n-1}) .$$

Now as $u_J = u_{I_n} - (u_{I_n} - u_J)$ and u_{I_n} is the classical level n combination we obtain the desired result. □

Now consider a sample $I_n \backslash \operatorname{supp}(U_{n,1})$, then $\underline{i} \in I_n \backslash \operatorname{supp}(U_{n,1})$ implies that $U_{\underline{i}} = 0$ if $|\underline{i}| = n$. As in the previous section we assume that time between failures is Weibull distributed with shape $0 < \kappa \leq 1$ and scale $\lambda > 0$ such that $\Pr(U_{\underline{i}} = 1) \leq 1 - e^{-(t_{\underline{i}}/\lambda)^\kappa}$. Using this we provide a point-wise bound for $\mathrm{E}[|u_{I_n \backslash \operatorname{supp}(U_{n,1})} - u|]$.

Proposition 5.9. *Fix the dimension $d = 2$ and $n \geq 0$. Let $u \in C([0,1]^d)$ and for each finite $J \subset \mathbb{N}^d$ let $u_J^{gcp} := \sum_{\underline{i} \in J} c_{\underline{i}} u_{\underline{i}}$ where the $c_{\underline{i}}$ are given by a solution of the* GCP *for the set J and the $u_{\underline{i}}$ satisfy the error splitting* (5.5) *with $p \geq 1$ and $K > 0$ such that $|\epsilon_x|, |\epsilon_y|, |\epsilon_{xy}| \leq K$ for $\boldsymbol{x}$ and $\underline{i}$. For each $u_{\underline{i}}$ let $t_{\underline{i}}$ be the time required to compute $u_{\underline{i}}$ and set $t_n = \max_{|\underline{i}|=n} t_{\underline{i}}$. Let $U_{n,s}$ be the random variable defined above, $s = 1$ and $\Pr(U_{\underline{i}} = 1) \leq 1 - e^{-(t_{\underline{i}}/\lambda)^\kappa}$ for each $\underline{i}$ with $\lambda > 0$ and $0 < \kappa \leq 1$. Then*

$$\mathrm{E}[|u^{gcp}_{I_n \backslash \operatorname{supp}(U_{n,1})} - u|] \leq \Big(3 + (1+2^p)n \\ + (n + 3 + (n-1)2^p)(1 - e^{-(t_n/\lambda)^\kappa})\Big) K (1+2^p) 2^{-2n} \,.$$

Proof. We observe that $u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,1})} = u_{I_n} - (u_{I_n} - u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,1})})$ and therefore via the triangle inequality

$$|u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,1})} - u| \leq |u_{I_n} - u| + |u_{I_n} - u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,1})}| \,.$$

For the right most term we observe that by Lemma 5.8

$$\begin{aligned} &u_{I_n} - u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,1})} \\ &\quad = \sum_{\substack{|\underline{i}|=n \\ \underline{i} \geq \underline{1}}} U_{\underline{i}} \Big((u_{i_1,i_2} - u) - (u_{i_1-1,i_2} - u) - (u_{i_1,i_2-1} - u) + (u_{i_1-1,i_2-1} - u) \Big) \\ &\qquad + U_{n,0} \left((u_{n,0} - u) - (u_{n-1,0} - u) \right) + U_{0,n} \left((u_{0,n} - u) - (u_{0,n-1} - u) \right) \,, \end{aligned}$$

and therefore by the triangle inequality

$$\begin{aligned} &|u_{I_n} - u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,1})}| \\ &\quad \leq \sum_{\substack{|\underline{i}|=n \\ \underline{i} \geq \underline{1}}} U_{\underline{i}} \left| (u_{i_1,i_2} - u) - (u_{i_1-1,i_2} - u) - (u_{i_1,i_2-1} - u) + (u_{i_1-1,i_2-1} - u) \right| \\ &\qquad + U_{n,0} \left| (u_{n,0} - u) - (u_{n-1,0} - u) \right| + U_{0,n} \left| (u_{0,n} - u) - (u_{0,n-1} - u) \right| \,. \end{aligned} \tag{5.6}$$

Substituting the error splitting (5.5) gives

$$\begin{aligned} &|(u_{i_1,i_2} - u) - (u_{i_1-1,i_2} - u) - (u_{i_1,i_2-1} - u) + (u_{i_1-1,i_2-1} - u)| \\ &\quad = \Big| \epsilon_{xy}(h_{i_1}, h_{i_2}) 2^{-np} - \epsilon_{xy}(h_{i_1-1}, h_{i_2}) 2^{-(n-1)p} \\ &\qquad - \epsilon_{xy}(h_{i_1}, h_{i_2-1}) 2^{-(n-1)p} + \epsilon_{xy}(h_{i_1-1}, h_{i_2-1}) 2^{-(n-2)p} \Big| \leq K(1+2^p)^2 2^{-np} \,, \end{aligned}$$

and

$$\begin{aligned}|(u_{n,0}-u)-(u_{n-1,0}-u)| &= \big|\epsilon_x(h_n)2^{-np}+\epsilon_{xy}(h_n,h_0)2^{-np}\\ &\quad -\epsilon_x(h_{n-1})2^{-(n-1)p}-\epsilon_{xy}(h_{n-1},h_0)2^{-(n-1)p}\big| \leq 2K(1+2^p)2^{-np}\,,\end{aligned}$$

and similarly for $|(u_{0,n}-u)-(u_{0,n-1}-u)|$. Substituting these bounds into (5.6) we take the expectation of both sides. As $\mathrm{E}[U_{\underline{i}}]=\Pr(U_{\underline{i}}=1)$ and expectation is both linear and monotone one obtains

$$\begin{aligned}\mathrm{E}\left[|u_{I_n}-u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}|\right] &\leq \sum_{\substack{|\underline{i}|=n\\ \underline{i}\geq\underline{1}}} \Pr(U_{\underline{i}}=1)K(1+2^p)^2 2^{-np}\\ &\qquad + (\Pr(U_{n,0}=1)+\Pr(U_{0,n}=1))2K(1+2^p)2^{-np}\\ &\leq (n+3+(n-1)2^p)\,(1-e^{-(t_n/\lambda)^\kappa})K(1+2^p)2^{-np}\,.\end{aligned}$$

Combining this with the bound $|u_{I_n}-u|\leq(3+(1+2^p)n)K2^{-np}$ (Lemma 2.25) yields the desired result. □

Note that as the right hand side is independent of $\boldsymbol{x}$ it also provides a bound for $\|\mathrm{E}[|u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}-u|]\|_\infty$.

We can extend the result of Proposition 5.9 to obtain a bound for the case $s=2$, that is when only those $u_{\underline{i}}$ with $|\underline{i}|\leq n-2$ are recomputed upon failure and the random vector $U_{n,2}$ reflects the state of those $u_{\underline{i}}$ with $n-2<|\underline{i}|\leq n$.

Proposition 5.10. *Fix $d=2$, $n\geq 0$ and consider the same assumptions as in Proposition 5.9 but with $s=2$. Additionally let $t_{n-1}=\max\{t_{\underline{i}}:|\underline{i}|=n-1\}$. Then*

$$\begin{aligned}\mathrm{E}[|u^{gcp}_{I_n\setminus\mathrm{supp}(U_{n,2})}-u|] &\leq \big(3+(1+2^p)n+(n+3+(n-1)2^p)(1+2^p)(1-e^{-(t_n/\lambda)^\kappa})\\ &\qquad (2n+2+3n2^p+(n-2)2^{2p})(1+2^p)(1-e^{-(t_{n-1}/\lambda)^\kappa})\big)K2^{-2n}\,.\end{aligned}$$

Proof. We may write $u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})}-u=(u_{I_n}-u)-(u_{I_n}-u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})})$ and thus

$$|u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})}-u|\leq|u_{I_n}-u|+|u_{I_n}-u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})}|\,.$$

For the right most term we observe that

$$\begin{aligned}|u_{I_n}-u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,2})}| &= \left|\sum_{\underline{i}\in\mathrm{supp}(U_{n,2})}(u_{i_1,i_2}-u_{i_1-1,i_2}-u_{i_1,i_2-1}+u_{i_1-1,i_2-1})\right|\\ &\leq \sum_{\underline{i}\in\mathrm{supp}(U_{n,2})}|u_{i_1,i_2}-u_{i_1-1,i_2}-u_{i_1,i_2-1}+u_{i_1-1,i_2-1}|\,,\end{aligned}$$

where $u_{i_1,i_2} := 0$ if $i_1 < 0$ and/or $i_2 < 0$. As in Proposition 5.5 given a sample of $I_n \backslash \operatorname{supp}(U_{n,2})$ we let J' be the largest downset contained in that sample. It follows that $\operatorname{supp}(U_{n,2}) \subseteq I_n \backslash J'$ and thus

$$|u_{I_n} - u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,2})}| \leq \sum_{\underline{i} \in I_n \backslash J'} |u_{i_1,i_2} - u_{i_1-1,i_2} - u_{i_1,i_2-1} + u_{i_1-1,i_2-1}| \,.$$

Now each term on the right hand side can be written as

$$|(u_{i_1,i_2} - u) - (u_{i_1-1,i_2} - u) - (u_{i_1,i_2-1} - u) + (u_{i_1-1,i_2-1} - u)|$$

for which we may substitute the error splitting and bound in the same way as in Proposition 5.9. Notice that now we have terms with $|\underline{i}| = n-1$ in addition to those with $|\underline{i}| = n$. In particular we also have the special cases $\underline{i} = (n-1, 0)$ and $\underline{i} = (0, n-1)$ for which $u_{i_1,i_2-1} = u_{i_1-1,i_2-1} = 0$ and $u_{i_1-1,i_2} = u_{i_1-1,i_2-1} = 0$ respectively. Now as in Proposition 5.5 we observe that $\underline{i} \in I_n \backslash J$ and $|\underline{i}| = n-1$ if and only if $u_{\underline{i}}$ failed, that is $U_{\underline{i}} = 1$. Further, $\underline{i} \in I_n \backslash J$ and $|\underline{i}| = n$ if and only if $u_{\underline{i}}$ failed ($U_{\underline{i}} = 1$) or $u_{\underline{i}-\underline{e}^k}$ failed ($U_{\underline{i}-\underline{e}^k} = 1$) for some $k \in \{1, 2\}$ (with $e^k_s = \delta_{k,s}$). Putting this together one obtains

$$\begin{aligned}
|u_{I_n} - u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,2})}| \leq & \sum_{\substack{|\underline{i}|=n-1 \\ \underline{i} \geq \underline{1}}} U_{\underline{i}} K (1+2^p)^2 2^{-(n-1)p} \\
& + (U_{n-1,0} + U_{0,n-1}) 2K(1+2^p) 2^{-(n-1)p} \\
& \sum_{\substack{|\underline{i}|=n \\ \underline{i} \geq \underline{1}}} \left(U_{\underline{i}} + \sum_{k=1}^{2} U_{\underline{i}-\underline{e}^k} \right) K(1+2^p)^2 2^{-np} \\
& + (U_{n,0} + U_{n-1,0} + U_{0,n} + U_{0,n-1}) 2K(1+2^p) 2^{-np} \,.
\end{aligned}$$

Now collecting the common $K(1+2^p)2^{-np}$, taking the expectation (which is linear and monotone) and noting $\mathrm{E}[U_{\underline{i}}] = \Pr(U_{\underline{i}} = 1)$ we have

$$\begin{aligned}
\mathrm{E}[|u_{I_n} - u^{\mathrm{gcp}}_{I_n \backslash \operatorname{supp}(U_{n,2})}|] \leq & K(1+2^p)2^{-np} \Bigg(\sum_{\substack{|\underline{i}|=n-1 \\ \underline{i} \geq \underline{1}}} \mathrm{E}[U_{\underline{i}}](1+2^p)2^p \\
& + (\mathrm{E}[U_{n-1,0}] + \mathrm{E}[U_{0,n-1}]) 2 \cdot 2^p \\
& + \sum_{\substack{|\underline{i}|=n \\ \underline{i} \geq \underline{1}}} \left(\mathrm{E}[U_{\underline{i}}] + \sum_{k=1}^{2} \mathrm{E}[U_{\underline{i}-\underline{e}^k}] \right) (1+2^p) \\
& + 2(\mathrm{E}[U_{n,0}] + \mathrm{E}[U_{n-1,0}] + \mathrm{E}[U_{0,n}] + \mathrm{E}[U_{0,n-1}]) \Bigg) \,.
\end{aligned}$$

We know that $\mathrm{E}[U_{\underline{i}}] = \Pr(U_{\underline{i}} = 1) \leq 1 - e^{-(t_{|\underline{i}|}/\lambda)^\kappa}$. Substituting this we find the sums no longer depend on the specific $\underline{i}$. Thus, as $\sum_{|\underline{i}|=n \,\&\, \underline{i} \geq \underline{1}} 1 = n-1$ and similarly for $|\underline{i}| = n-1$, we obtain

$$\begin{aligned}
\mathrm{E}[|u_{I_n} - u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,2})}|] \\
&\leq K(1+2^p)2^{-2n}\Big((n-2)(1-e^{-(t_{n-1}/\lambda)^\kappa})2^p(1+2^p) + 4(1-e^{-(t_{n-1}/\lambda)^\kappa})2^p \\
&\qquad + (n-1)((1-e^{-(t_n/\lambda)^\kappa}) + 2(1-e^{-(t_{n-1}/\lambda)^\kappa}))(1+2^p) \\
&\qquad + 4(1-e^{-(t_n/\lambda)^\kappa}) + 4(1-e^{-(t_{n-1}/\lambda)^\kappa})\Big) \\
&= K(1+2^p)2^{-2n}\Big((2n+2+3n2^p+(n-2)2^{2p})(1-e^{-(t_{n-1}/\lambda)^\kappa}) \\
&\qquad + (n+3+(n-1)2^p)(1-e^{-(t_n/\lambda)^\kappa})\Big) .
\end{aligned}$$

Adding to this the bound $|u_{I_n} - u| \leq (3 + (1+2^p)n)K2^{-np}$ (Lemma 2.25) yields the desired result. □

Similar to Proposition 5.5 this result is an over-estimate of the expected error but gives a rough idea of how the error depends on the distribution of the time between failures. It should be clear how the technique used to prove Proposition 5.10 could be applied to more general $U_{n,s}$ but for $s > 2$ the resulting bound becomes quite weak for the same reasons as discussed for Proposition 5.5. Observe that Propositions 5.9 and 5.10 only apply to $d = 2$. The main difficulty in extending this to $d > 2$ is that there are more cases of indices $\underline{i}$ for which some of the $\underline{i} - \underline{j}$ with $\underline{0} \leq \underline{j} \leq \underline{1}$ have negative components. These affect the error splitting differently depending on the number of non-zero components as we will see in the following lemma.

Lemma 5.11. *Let $d \in \mathbb{N}_+$ and $\underline{i} \in \mathbb{N}^d$. For $\underline{0} \leq \underline{j} \leq \underline{1}$ consider approximations $u_{\underline{i}-\underline{j}}$ of some function u which satisfy the error splitting*

$$u_{\underline{i}} - u = \sum_{k=1}^{d} \sum_{\substack{\{e_1,\dots,e_k\} \\ \subset \{1,\dots,d\}}} \gamma_{e_1,\dots,e_k}(h_{i_{e_1}}, \dots, h_{i_{e_k}}) h^p_{i_{e_k}} \cdots h^p_{i_{e_k}} , \tag{5.7}$$

with $|\gamma_{e_1,\dots,e_k}| \leq K$ for all $\{e_1,\dots,e_k\} \subset \{1,\dots,d\}$ and $\boldsymbol{x}$ for some $K > 0$. Additionally, let $u_{\underline{i}-\underline{j}} = 0$ if $i_k - j_k < 0$ for any $k = 1,\dots,d$. Then

$$\left| \sum_{\underline{0} \leq \underline{j} \leq \underline{1}} (-1)^{|\underline{j}|} u_{\underline{i}-\underline{j}} \right| \leq (1+2^p)^{d-\sigma} K 2^{-p|\underline{i}|} 2^\sigma .$$

where $\sigma = d - |\underline{i}|_0$ is the number of zero components of $\underline{i}$.

Proof. Let $\{l_1,\dots,l_\sigma\}\subset\{1,\dots,d\}$ be such that $i_{l_1}=\cdots=i_{l_\sigma}=0$. Without loss of generality we assume that $l_i=i$ for $i=1,\dots,\sigma$. It follows that

$$\begin{aligned}\sum_{\underline{0}\le\underline{j}\le\underline{1}}(-1)^{|\underline{j}|}u_{\underline{i}-\underline{j}} &= \sum_{j_{\sigma+1}=0}^{1}\cdots\sum_{j_d=0}^{1}(-1)^{j_{\sigma+1}+\cdots+j_d}u_{(i_1,\dots,i_\sigma,i_{\sigma+1}-j_{\sigma+1},\dots,i_d-j_d)}\\ &= \sum_{j_{\sigma+1}=0}^{1}\cdots\sum_{j_d=0}^{1}(-1)^{j_{\sigma+1}+\cdots+j_d}(u_{(i_1,\dots,i_\sigma,i_{\sigma+1}-j_{\sigma+1},\dots,i_d-j_d)}-u)\,.\end{aligned}$$

Now we may substitute the error splitting into this last equality. First consider substituting just one of the terms from the error splitting into this equation, that is fix $k\in\{1,\dots,d\}$ and $\{e_1,\dots,e_k\}\subset\{1,\dots,d\}$ and consider substituting

$$u_{(i_1,\dots,i_\sigma,i_{\sigma+1}-j_{\sigma+1},\dots,i_d-j_d)}-u=\gamma_{e_1,\dots,e_k}(h_{i_{e_1}-j_{e_1}},\dots,h_{i_{e_k}-j_{e_k}})h^p_{i_{e_1}-j_{e_1}}\cdots h^p_{i_{e_k}-j_{e_k}}\,,$$

where $j_1=\cdots=j_\sigma=0$. Observe that

$$\sum_{j_{\sigma+1}=0}^{1}\cdots\sum_{j_d=0}^{1}(-1)^{j_{\sigma+1}+\cdots+j_d}\gamma_{e_1,\dots,e_k}(h_{i_{e_1}-j_{e_1}},\dots,h_{i_{e_k}-j_{e_k}})h^p_{i_{e_1}-j_{e_1}}\cdots h^p_{i_{e_k}-j_{e_k}}$$

is zero whenever $\{\sigma+1,\dots,d\}\cap(\{1,\dots,d\}\backslash\{e_1,\dots,e_k\})$ is non-empty. Thus an error term does not experience cancellation if this set is empty, or equivalently if $\{l_{\sigma+1},\dots,l_d\}\subseteq\{e_1,\dots,e_k\}$. Now fixing a $\{l_{\sigma+1},\dots,l_d\}\subseteq\{e_1,\dots,e_k\}$ then without loss of generality we let $\{l_{\sigma+1},\dots,l_{\sigma+m}\}=\{l_{\sigma+1},\dots,l_d\}\cap\{e_1,\dots,e_k\}$. We observe that

$$\begin{aligned}&\left|\sum_{j_{\sigma+1}=0}^{1}\cdots\sum_{j_d=0}^{1}(-1)^{j_{\sigma+1}+\cdots+j_d}\gamma_{e_1,\dots,e_k}(h_{i_{e_1}-j_{e_1}},\dots,h_{i_{e_k}-j_{e_k}})h^p_{i_{e_1}-j_{e_1}}\cdots h^p_{i_{e_k}-j_{e_k}}\right|\\ &=\left|\sum_{j_{\sigma+1}=0}^{1}\cdots\sum_{j_{\sigma+m}=0}^{1}(-1)^{j_{\sigma+1}+\cdots+j_{\sigma+m}}\gamma_{e_1,\dots,e_k}(h_{i_{e_1}-j_{e_1}},\dots,h_{i_{e_k}-j_{e_k}})\right.\\ &\qquad\qquad\left.h^p_{i_{e_1}-j_{e_1}}\cdots h^p_{i_{e_k}-j_{e_k}}\right|\le K(1+2^p)^m h^p_{i_{e_1}}\cdots h^p_{i_{e_k}}\,.\end{aligned}$$

Further, as $\{l_{\sigma+1},\dots,l_d\}\subseteq\{e_1,\dots,e_k\}$ then $i_{e_1}+\cdots+i_{e_k}=|\underline{i}|$ and therefore $h^p_{i_{e_1}}\cdots h^p_{i_{e_k}}=h^p_{|\underline{i}|}$. Having now established a bound when a single term of the error splitting is substituted we now wish to sum over all of the error splitting terms. Note that for each $k=1,\dots,d$, the number of $\{e_1,\dots,e_k\}\subset\{1,\dots,d\}$ which contain $\{l_{\sigma+1},\dots,l_d\}$ is equivalent to the number of $\{l_1,\dots,l_\sigma\}$ which contain

$\{1,\dots,d\}\backslash\{e_1,\dots,e_k\}$ which is clearly $\binom{\sigma}{d-k}$. Now summing over $k=d-\sigma,\dots,d$ we obtain

$$\left|\sum_{\underline{0}\le\underline{j}\le\underline{1}}(-1)^{|\underline{j}|}u_{\underline{i}-\underline{j}}\right| \le \sum_{k=d-\sigma}^{d}\binom{\sigma}{d-k}K(1+2^p)^{d-\sigma}h^p_{|\underline{i}|}$$
$$= 2^\sigma K(1+2^p)^{d-\sigma}2^{-p|\underline{i}|}\,,$$

as required. □

This lemma allows us to give a result for $U_{n,1}$ for general $d\ge 2$.

Proposition 5.12. *Fix the dimension $d\ge 2$ and $n\ge 0$. Let $u\in C([0,1]^d)$ and for each finite $J\subset\mathbb{N}^d$ let $u_J^{gcp}:=\sum_{\underline{i}\in J}c_{\underline{i}}u_{\underline{i}}$ where the $c_{\underline{i}}$ are given by a solution of the* GCP *for the set J and the $u_{\underline{i}}$ satisfy the error splitting (5.7) with $p\ge 1$ and $K>0$ such that such that each $|\gamma_{e_1,\dots,e_k}|\le K$. For each $u_{\underline{i}}$ let $t_{\underline{i}}$ be the time required to compute $u_{\underline{i}}$ and set $t_n=\max_{|\underline{i}|=n}t_{\underline{i}}$. Let $U_{n,s}$ be the random variable defined previously, $s=1$ and $\Pr(U_{\underline{i}}=1)\le 1-e^{-(t_{\underline{i}}/\lambda)^\kappa}$ for each $\underline{i}$ with $\lambda>0$ and $0<\kappa\le 1$. Then,*

$$\mathrm{E}[|u^{gcp}_{I_n\backslash\operatorname{supp}(U_{n,1})}-u|]\le K2^{-pn}(1+2^p)^{d-1}\left(\binom{n+2d-1}{d-1}\right.$$
$$\left.+(1-e^{-(t_n/\lambda)^\kappa})(1+2^p)\sum_{\sigma=0}^{d-1}\binom{n-1}{d-1-\sigma}\binom{d}{\sigma}\left(\frac{2}{1+2^p}\right)^\sigma\right).$$

Proof. The triangle inequality yields

$$|u^{\mathrm{gcp}}_{I_n\backslash\operatorname{supp}(U_{n,1})}-u|\le|u_{I_n}-u|+|u_{I_n}-u^{\mathrm{gcp}}_{I_n\backslash\operatorname{supp}(U_{n,1})}|\,.$$

The first term is bounded above by Theorem 2.27, specifically

$$|u_{I_n}-u|\le K2^{-pn}(1+2^p)^{d-1}\binom{n+2d-1}{d-1}\,.$$

For the latter term we observe that

$$u_{I_n}-u^{\mathrm{gcp}}_{I_n\backslash\operatorname{supp}(U_{n,1})}=\sum_{\underline{i}\in\operatorname{supp}(U_{n,1})}\sum_{\underline{0}\le\underline{j}\le\underline{1}}(-1)^{|\underline{j}|}u_{\underline{i}-\underline{j}}\,, \tag{5.8}$$

(with $u_{\underline{i}-\underline{j}}:=0$ if $i_k-j_k<0$ for any $k\in\{1,\dots,d\}$). To bound the right hand side we may use Lemma 5.11 but first we must split this sum up over the $\underline{i}$ with different number of non-zero components. One has

$$\begin{aligned}|u_{I_n} - u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}| &= \left|\sum_{\sigma=0}^{d-1}\sum_{\substack{\underline{i}\in\mathrm{supp}(U_{n,1})\\|\underline{i}|_0=d-\sigma}}\sum_{\underline{0}\le\underline{j}\le\underline{1}}(-1)^{|\underline{j}|}u_{\underline{i}-\underline{j}}\right| && \text{by (5.8)}\\
&\le\sum_{\sigma=0}^{d-1}\sum_{\substack{\underline{i}\in\mathrm{supp}(U_{n,1})\\|\underline{i}|_0=d-\sigma}}\left|\sum_{\underline{0}\le\underline{j}\le\underline{1}}(-1)^{|\underline{j}|}u_{\underline{i}-\underline{j}}\right| && \text{tri. ineq.}\\
&\le\sum_{\sigma=0}^{d-1}\sum_{\substack{\underline{i}\in\mathrm{supp}(U_{n,1})\\|\underline{i}|_0=d-\sigma}}2^\sigma K(1+2^p)^{d-\sigma}2^{-p|\underline{i}|} && \text{Lem. 5.11.}\end{aligned}$$

Now we note that the size of the set $|\{\underline{i}\in\mathrm{supp}(U_{n,1}) : |\underline{i}|_0 = d-\sigma\}|$ is $\binom{n-1}{d-1-\sigma}\binom{d}{\sigma}$ as for a given σ there are $\binom{d}{\sigma}$ partitions $\{l_1,\dots,l_\sigma\}\cup\{l_{\sigma+1},\dots,l_d\}=\{1,\dots,d\}$ and for each of these there are $\binom{n-1}{d-1-\sigma}$ different $\{i_{l_{\sigma+1}},\dots,i_{l_d}\}\in\mathbb{N}_+^{d-\sigma}$ which sum to n. Further, $\underline{i}$ is in $\mathrm{supp}(U_{n,1})$ if and only if $U_{\underline{i}}=1$. As $\Pr(U_{\underline{i}}=1)\le 1-e^{-(t_{|\underline{i}|}/\lambda)^\kappa}$ one obtains

$$\mathrm{E}[|u_{I_n} - u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}|]\le\sum_{\sigma=0}^{d-1}\binom{n-1}{d-1-\sigma}\binom{d}{\sigma}(1-e^{-(t_n/\lambda)^\kappa})2^\sigma K(1+2^p)^{d-\sigma}2^{-pn}\,.$$

Adding to this the bound for $|u_{I_n}-u|$ we obtain the desired result. □

We can sacrifice some tightness in this result to simplify the expression. As $\frac{2}{1+2^p}\le 1$ and $\sum_{\sigma=0}^{d-1}\binom{n-1}{d-1-\sigma}\binom{d}{\sigma}=\binom{n+d-1}{d-1}$ via Vandermonde's identity we have

$$\begin{aligned}\mathrm{E}[|u_{I_n} - u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}|] &\le\binom{n+d-1}{d-1}(1-e^{-(t_n/\lambda)^\kappa})K(1+2^p)^d2^{-pn}\\
&\le\binom{n+2d-1}{d-1}(1-e^{-(t_n/\lambda)^\kappa})K(1+2^p)^d2^{-pn}\,.\end{aligned}$$

Adding this to the bound for $|u_{I_n}-u|$ then yields

$$\begin{aligned}&\mathrm{E}[|u^{\mathrm{gcp}}_{I_n\setminus\mathrm{supp}(U_{n,1})}-u|]\\
&\qquad\le\big(1+(1+2^p)(1-e^{-(t_n/\lambda)^\kappa})\big)\binom{n+2d-1}{d-1}K2^{-pn}(1+2^p)^{d-1}\,.\end{aligned}$$

The result of Proposition 5.12 can be extended to $U_{n,2}$ in much the same way that Proposition 5.9 was extended to Proposition 5.10. As it should be clear how this is done at this point we state the result in the weaker but simpler form without proof.

Proposition 5.13. *Fix $d \geq 2$, $n \geq 0$ and consider the same assumptions as in Proposition 5.12 but with $s = 2$. Then*

$$\begin{aligned}\mathrm{E}[|u^{gcp}_{I_n \setminus \mathrm{supp}(U_{n,2})} - u|] \leq \Big(1 + (1+2^p)\left(1 - e^{-(t_n/\lambda)^\kappa} + d(1 - e^{-(t_{n-1}/\lambda)^\kappa})\right) \\ + 2^p(1+2^p)(1 - e^{-(t_{n-1}/\lambda)^\kappa})\Big)\binom{n+2d-1}{d-1} K(1+2^p)^{d-1}2^{-pn}\,.\end{aligned}$$

Thus far we have studied the expectation of pointwise error estimates. As with the interpolation estimates in Section 5.2.2 it would also be useful to know something about the variance. There is no clear way that one go about estimating $\mathrm{E}[(u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,1})} - \mathrm{E}[u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,1})}])^2]$ but as we expect $\mathrm{E}[u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,1})}]$ to be close to u_{I_n} for small fault rates then we will instead study $\mathrm{E}[(u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,1})} - u_{I_n})^2]$.

Proposition 5.14. *Fix $d \geq 2$, $n \geq d-1$ and consider the same assumptions as in Proposition 5.12. Then*

$$\begin{aligned}\mathrm{E}[(u^{gcp}_{I_n \setminus \mathrm{supp}(U_{n,1})} - u_{I_n})^2] \leq K^2(1+2^p)^{2d}2^{-2pn}\left(\binom{n+d-1}{d-1}(1 - e^{-(t_n/\lambda)^\kappa})\right. \\ \left. + \binom{n+d-1}{d-1}\left(\binom{n+d-1}{d-1} - 1\right)(1 - e^{-(t_n/\lambda)^\kappa})^2\right)\,.\end{aligned}$$

Proof. From the proof of Proposition 5.12 we observe that

$$\begin{aligned}|u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,1})} - u_{I_n}| &\leq \sum_{\underline{i} \in \mathrm{supp}(U_{n,1})} K(1+2^p)^d 2^{-pn} \\ &= \sum_{|\underline{i}|=n} U_{\underline{i}} K(1+2^p)^d 2^{-pn}\,.\end{aligned}$$

Therefore

$$(u^{\mathrm{gcp}}_{I_n \setminus \mathrm{supp}(U_{n,1})} - u_{I_n})^2 \leq K^2(1+2^p)^{2d}2^{-2pn}\left(\sum_{\substack{|\underline{i}|=|\underline{j}|=n \\ \underline{i} \neq \underline{j}}} U_{\underline{i}}U_{\underline{j}} + \sum_{|\underline{i}|=n} U_{\underline{i}}^2\right)\,.$$

Upon taking the expectation we use the fact that

$$\mathrm{E}\left[\sum_{\substack{|\underline{i}|=|\underline{j}|=n \\ \underline{i} \neq \underline{j}}} U_{\underline{i}}U_{\underline{j}}\right] \leq \binom{n+d-1}{d-1}\left(\binom{n+d-1}{d-1} - 1\right)(1 - e^{-(t_n/\kappa)^\kappa})^2$$

and similarly

$$\mathrm{E}\left[\sum_{|\underline{i}|=n} U_{\underline{i}}^2\right] \leq \binom{n+d-1}{d-1}(1 - e^{-(t_n/\kappa)^\kappa})$$

to obtain the desired result. □

It is clear how this could be extended to the case $U_{n,2}$. This concludes our error estimates for the FTCT and we now proceed with some numerical results.

5.2.4 Numerical results

In this section, we present numerical results for the FTCT, specifically Algorithm 3, applied to numerical solutions of the advection equation using simulated faults (as described in Section 5.2.1). In particular, we demonstrate that the FTCT has low overhead and that the expected error is close to the error obtained in the absence of faults. The performance of the FTCT has been studied in several papers [65, 66, 3, 69]. We point out that the FTCT has been successfully tested on more complex problems, for example results for the application GENE which performs gyrokinetic plasma turbulence simulations are reported in [5, 72], but these will not be reported in this thesis. Here we focus on the scalar advection equation for which the exact solution is known so that we can compute the error for each of our computations. Many of the results and conclusions presented in this section are taken from the paper [69] and an earlier preprint.

As in Section 4.5, the PDE we choose to test the FTCT on is the scalar advection equation

$$\frac{\partial u}{\partial t} + \boldsymbol{a} \cdot \nabla u = 0 \, .$$

Here we consider problems in 3 spatial dimensions on the domain $[0,1]^3 \subset \mathbb{R}^3$ with flow field $\boldsymbol{a} \in \mathbb{R}^3$. We consider two different variations of this problem. The first considers a constant $\boldsymbol{a} = (1,1,1)$ whilst the second considers a $\boldsymbol{a}$ which depends on the first two spatial coordinates. We compute approximations of $u(\boldsymbol{x}, t)$ on the grids $\Omega_{\underline{i}} = \{j_1 2^{-i_1} : j_1 = 0,1,\dots,2^{i_1}\} \times \{j_2 2^{-i_2} : j_2 = 0,1,\dots,2^{i_2}\} \times \{j_3 2^{-i_3} : j_1 = 0,1,\dots,2^{i_3}\}$ for $\underline{i} \in \mathbb{N}^d$. For a fixed solution time t we denote the approximation of u computed on $\Omega_{\underline{i}}$ by $u_{\underline{i}}$.

A truncated combination technique as in (4.1) is used as the starting point for our experiments. This is because the combination technique is known to give poor results for some problems when some of the grids are too coarse in one or more dimensions. A reason for this is that the initial condition may be poorly represented on such grids. This can lead to solutions that differ significantly from the exact solution which do not extrapolate in the way one usually expects. In order to apply the FTCT we need to compute some additional grids which are used in the event of failures. We define

$$I_{n,\tau} := \{\underline{i} \in \mathbb{N}^d : \min\{i_1, \dots, i_d\} \geq \tau \text{ and } n - d - 1 \leq |\underline{i}| \leq n\}$$

which is the set of indices for which we are required to compute solutions $u_{\underline{i}}$ (that is approximations to u computed on the grid) if the top two levels are not to be recomputed in the event of a fault. The corresponding random vector denoting the failure or success of $u_{\underline{i}}$ which are not recomputed upon failure is

$$U_{n,\tau,s} := (U_{\underline{i}})_{\underline{i} \in I_{n,\tau} \backslash I_{n-s,\tau}} \,,$$

with $s = 2$. We compute samples of the resulting $u^{\text{gcp}}_{I_{n,\tau} \backslash \, \text{supp}(U_{n,\tau,s})}$ via an implementation of the FTCT with fault simulation as described in Algorithm 3 with each $u_{\underline{i}}$ being a finite difference solution to an advection PDE.

Constant flow field

Here we consider constant $\boldsymbol{a} = (1,1,1)$, periodic boundary conditions and the initial condition

$$u(\boldsymbol{x}, 0) = \sin(4\pi x_1) \sin(2\pi x_2) \sin(2\pi x_3) \,.$$

Notice that the initial condition itself is periodic and is consistent with the periodic boundary condition, in particular $u(\boldsymbol{x} + i\boldsymbol{e}_1 + j\boldsymbol{e_2} + k\boldsymbol{e}_3, 0) = u(\boldsymbol{x}, 0)$ where $i, j, k \in \mathbb{Z}$ and $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$ are the standard unit basis vectors $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ respectively. Via the method of characteristics it is straightforward to show that the exact solution is given by

$$u(x,t) = \sin(4\pi(x_1 - a_1 t)) \sin(2\pi(x_2 - a_2 t)) \sin(2\pi(x_3 - a_3 t)) \,.$$

The PDE is solved numerically using a tensor product application of the Lax-Wendroff finite difference scheme (that is we apply (3.12) over each dimension) giving results which are second order in space and time. The time step size must be sufficiently small such that the stability criterion is satisfied for update in each direction.

Note that as the grid sizes vary between the $u_{\underline{i}}$ so does the maximum stable time step size as determined by the CFL condition. We choose the same time step size for all component solutions to avoid instability that may otherwise arise from the extrapolation of time stepping errors during the combination. As a result our time steps must satisfy the CFL condition for all component grids. By choosing Δt such that it satisfies the CFL condition for the numerical solution of $u_{(n-2\tau, n-2\tau, n-2\tau)}$ it follows that the CFL condition is also satisfied for all $u_{\underline{i}}$ with $\underline{i} \in I_{n,\tau}$.

Table 5.1: *Numerical results for $r = 100$ samples of $u^{gcp}_{I_{n,\tau}\setminus \mathrm{supp}(U_{n,2})}$ for different n, τ using the Weibull distribution with mean of 1000 seconds and shape parameter of 0.7 for the fault simulation. The computations were performed on 2 nodes with 6 OpenMP threads on each.*

n	τ	f_{ave}	ϵ_{ave}	ϵ_{min}	ϵ_{max}	w_{ave}	w_{min}	w_{max}
18	4	0.13	2.103e-4	2.096e-4	2.298e-4	26.96	26.94	27.07
20	5	0.37	7.064e-5	7.004e-5	8.580e-5	72.03	71.82	72.72
19	4	0.48	6.355e-5	6.266e-5	6.899e-5	131.1	130.8	131.9
21	5	1.21	1.979e-5	1.886e-5	3.030e-5	379.9	379.3	381.9
20	4	1.80	1.925e-5	1.856e-5	2.156e-5	649.8	648.2	653.3

All the computations for this problem were performed on a Fujitsu PRIMERGY cluster provided by Fujitsu Laboratories of Europe. This machine consists of 36 nodes, each with 2 Intel Xeon X5670 CPUs (6 core, 2.934GHz) and Infiniband interconnect.

We first looked at the effect of simulated faults on the error of the computed solution. Given level n and truncation parameter τ the code was executed for some number of runs r on a fixed number of nodes using the same number of threads. Component solutions are combined twice in each run, once halfway through the time steps and again at the end. For each run we recorded the number of faults f that occurred, the l_1 error of the solution ϵ and the wall time w spent in the solver (we exclude some initial startup overhead and the error calculation from the timing). We then calculated the average number of faults

$$f_{ave} = \frac{1}{r}\sum_{k=1}^{r} f_k \,,$$

the average, minimal and maximal observed errors

$$\epsilon_{ave} = \frac{1}{r}\sum_{k=1}^{r} \epsilon_k \,, \quad \epsilon_{min} = \min\{\epsilon_1, \dots, \epsilon_r\} \,, \quad \epsilon_{max} = \max\{\epsilon_1, \dots, \epsilon_r\} \,,$$

and the average, minimal and maximal observed wall times (in seconds)

$$w_{ave} = \frac{1}{r}\sum_{k=1}^{r} w_k \,, \quad w_{min} = \min\{w_1, \dots, w_r\} \,, \quad w_{max} = \max\{w_1, \dots, w_r\} \,.$$

Table 5.1 shows our results for $r = 100$ runs of the FTCT with fault simulation on 2 nodes with 6 OpenMP threads on each. Faults were simulated as described

Table 5.2: *Numerical results for $r = 200$ samples of $u^{gcp}_{I_{n,\tau}\setminus\text{supp}(U_{n,2})}$ for each n, τ using the Weibull distribution with mean of 1000 seconds and shape parameter of 0.7 for the fault simulation. The computations were performed on 6 nodes with 6 OpenMP threads on each.*

n	τ	f_{ave}	ϵ_{ave}	ϵ_{min}	ϵ_{max}	w_{ave}	w_{min}	w_{max}
18	4	0.305	2.119e-4	2.096e-4	2.418e-4	9.216	9.196	9.392
20	5	0.535	7.166e-5	7.004e-5	1.170e-4	24.77	24.67	25.22
19	4	0.690	6.385e-5	6.244e-5	7.392e-5	44.55	44.40	45.90
21	5	1.805	2.015e-5	1.886e-5	3.179e-5	131.7	130.9	134.6
20	4	2.475	1.961e-5	1.844e-5	2.478e-5	224.4	223.3	228.7

in Section 5.2.1 using the Weibull distribution with mean of 1000 seconds and shape parameter 0.7 to sample the time between failures. As we increase the level n (or decrease τ) we increase the problem size and hence computation time. This in turn leads to an increase in the average number of faults that occur per run as seen in the f_{ave} column. The minimal error is the same as the error without failure (sometimes it is fractionally smaller). Comparing with the average error we see that the additional error generated by recovery from simulated faults is small. Also worth noting is that the variability in computation time is quite small indicating that any recomputations, when they occur, do not seem to cause any significant disruptions.

In Table 5.2 we repeat this experiment with $r = 200$ runs on 6 nodes with 6 OpenMP threads on each. Whilst running with additional nodes leads to a decrease in computation time we experience more faults on average because of the additional nodes. However, we can see that the effect of the increased average number of faults is quite small on both the average solution error and the average wall time.

Table 5.3 again shows results for $r = 100$ runs of the FTCT with fault simulation on 2 nodes with 6 OpenMP threads on each. However, for this experiment the faults are exponentially distributed with a mean of 1000 seconds. We see that for this distribution the faults are a little less frequent on average leading to a slightly smaller average error. Similar is observed in Table 5.4 where we repeat the experiment with $r = 200$ runs on 6 nodes with 6 OpenMP threads on each. Here the average number of faults is substantially less than the results of Table 5.2 and this is again reflected by a smaller average error in comparison. The large w_{max} in the 2nd row is due to a single outlier, the next largest time

Table 5.3: *Numerical results for* $r = 100$ *samples of* $u^{gcp}_{I_{n,\tau}\setminus\text{supp}(U_{n,2})}$ *for each* l, τ *using exponentially distributed time between simulated failures with mean of* 1000 *seconds. Computations were performed on 2 nodes with 6* OpenMP *threads on each.*

l	τ	f_{ave}	ϵ_{ave}	ϵ_{min}	ϵ_{max}	w_{ave}	w_{min}	w_{max}
18	4	0.06	2.098e-4	2.096e-4	2.230e-4	26.96	26.93	27.10
20	5	0.17	7.098e-5	7.006e-5	1.155e-4	72.01	71.87	72.47
19	4	0.26	6.321e-5	6.266e-5	7.283e-5	131.1	130.8	131.7
21	5	0.71	1.942e-5	1.886e-5	2.980e-5	379.8	379.3	380.8
20	4	1.27	1.921e-5	1.856e-5	2.127e-5	649.3	647.9	653.4

Table 5.4: *Numerical results for* $r = 100$ *samples of* $u^{gcp}_{I_{n,\tau}\setminus\text{supp}(U_{n,2})}$ *for each* l, τ *with exponentially distributed time between simulated failures with mean of* 1000 *seconds. Computations were performed on 6 nodes with 6 OpenMP threads on each.*

l	τ	f_{ave}	ϵ_{ave}	ϵ_{min}	ϵ_{max}	w_{ave}	w_{min}	w_{max}
18	4	0.070	2.103e-4	2.096e-4	2.296e-4	9.231	9.214	9.340
20	5	0.155	7.034e-5	7.004e-5	8.162e-5	24.80	24.70	30.83
19	4	0.265	6.327e-5	6.244e-5	7.039e-5	44.48	44.36	45.07
21	5	0.865	1.936e-5	1.886e-5	3.203e-5	131.7	131.0	134.5
20	4	1.415	1.921e-5	1.844e-5	2.178e-5	224.0	222.9	227.4

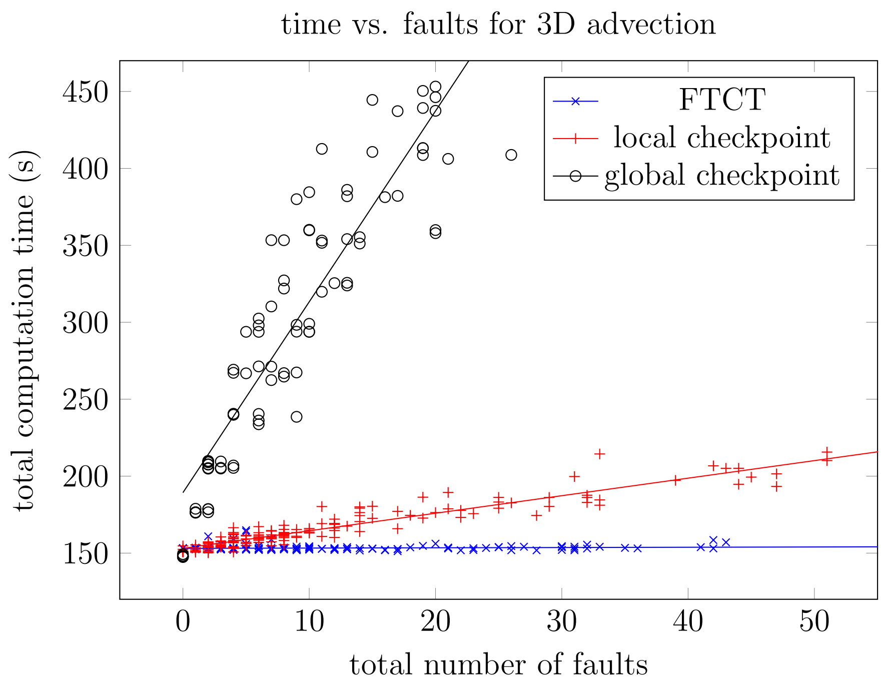


Figure 5.1: *We compare the time taken to compute the solution to the 3D advection problem using three different approaches to fault tolerance. The problem size is fixed at level* $n = 21$ *with truncation parameter* $\tau = 5$*. All computations used 6* MPI *processes with 6 OpenMP threads each. Component solutions are combined 4 times throughout the computation and it is during the combination that we check for faults. For each method the problem was run numerous times with* MTTF *varying from* 25 *to* 1000 *seconds.*

being 25.34. No simulated faults occurred for this outlier so we suspect it was due to a system issue.

In Figure 5.1 we compare the computation time required for our approach to reach a solution compared to more traditional checkpointing approaches, in particular, with a local and global checkpointing approaches described by Algorithms 2 and 1 respectively. Note that for the checkpointing methods the extra component solutions required for the FTCT are not required and are hence not computed. As a result these approaches are slightly faster when no faults occur, although the difference is small. As the number of faults increases, it can be seen from Figure 5.1 that the computation time for the local and global checkpointing methods begins to grow. A line of best fit has been added to the figure which makes it clear that the time for recovery with global checkpointing increases rapidly with the number of faults. Local checkpointing is a significant improvement on this but still shows some growth. On the other hand our approach

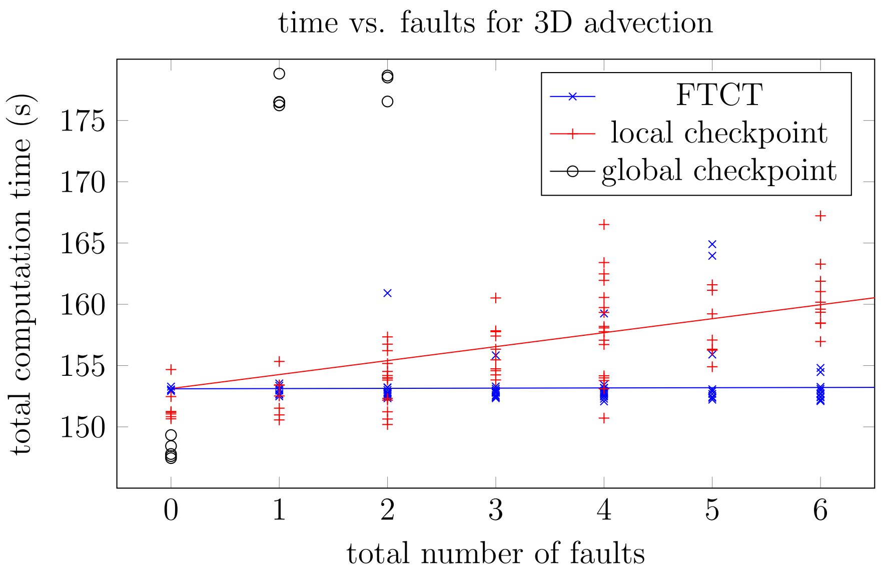


Figure 5.2: *A zoom of the bottom left corner of Figure 5.1. It is clearer that the* FTCT *approach takes slightly longer when no faults occur. With one fault the run times of the* FTCT *method have much less spread and are slightly better on average. For two or more faults the advantage is much clearer.*

is barely affected by the number of faults and outperforms both the local and global checkpointing approaches after only a few faults. For much larger number of faults our approach is significantly better. It is important to point out that we tuned neither of the checkpointing or FTCT algorithms to be optimal with respect to the mean time between failures (MTBFs). The compute-checkpoint interval was the same for all methods and all MTBFs.

Divergence free flow field

In this section the advection equation is solved with the divergence free velocity field

$$\boldsymbol{a}(\boldsymbol{x}) = (\sin(\pi x_1)\cos(\pi x_2), -\cos(\pi x_1)\sin(\pi x_2), 1)\,,$$

and initial condition

$$u(\boldsymbol{x},0) = \exp\left(-\frac{\pi^2}{2}\left(x_1-\frac{3}{8}\right)^2 - 2\pi^2\left(x_2-\frac{3}{8}\right)^2 - 2(1+\cos(2\pi x_3))\right)\,.$$

Notice that the flow field is such that x_1 and x_2 boundary conditions are not required. In the x_3 direction we specify the boundary condition $u|_{x_3=1} = u|_{x_3=0}$. The initial condition is a Gaussian like peak centred around $\left(\frac{3}{8},\frac{3}{8},\frac{1}{2}\right)$ which flows

Table 5.5: *Numerical results for $r = 50$ samples of $u^{gcp}_{I_{n,\tau}\setminus\mathrm{supp}(U_{n,2})}$ with level $n = 22$ and truncation $\tau = 5$ (which computes 110 different $u_{\underline{i}}$). Faults were simulated with varying* MTBF *(in seconds) to study the effect on error and time to solution. The simulated faults are Weibull distributed with shape $\kappa = 0.7$ in each case. Component solutions were combined 4 times throughout the computation. The computations were performed on 64 processors (8* MPI *processes with 8 OpenMP threads each). Times reported are for the inner computation loop and exclude overheads (including the load balancing and error calculations).*

MTBF	f_{ave}	ϵ_{ave}	ϵ_{min}	ϵ_{max}	w_{ave}	w_{min}	w_{max}
3600	1.22	1.132e-5	1.057e-5	1.815e-5	152.1	151.4	153.2
1800	1.78	1.159e-5	1.057e-5	1.848e-5	153.0	151.9	155.7
900	3.48	1.293e-5	1.057e-5	2.044e-5	150.9	149.0	153.5
450	5.66	1.412e-5	1.058e-5	2.048e-5	152.3	151.0	153.8
225	9.42	1.641e-5	1.059e-5	3.199e-5	153.0	151.4	155.0
112.5	16.00	1.803e-5	1.064e-5	3.452e-5	152.9	151.0	155.4
56.25	26.76	2.170e-5	1.159e-5	3.593e-5	154.2	152.0	157.9

in a helix like pattern around the line $x_1 = x_2 = 1/2$. (The flow in the x_1, x_2 coordinates is the same as that considered in Section 4.5.2). We evolve from $t = 0$ up to $t = 0.25$ in our computations using a second order central difference discretisation of spatial derivatives and fourth order Runge–Kutta (RK4) time integration (i.e. analogous to that described for the two dimensional advection problem in Section 4.5.1). We compare numerical solutions against the exact solution obtained via the method of characteristics computed with an RK4 solver.

Similar to the constant flow problem, we use the same Δt for the computation of all $u_{\underline{i}}$ and thus Δt must be sufficiently small such that all of these are stable. All of our computations for this problem were performed using the Raijin cluster at the National Computational Infrastructure[2] which is supported by the Australian Government. Raijin is a Fujitsu PRIMERGY cluster consisting of 3592 compute nodes, each with two Intel Xeon Sandy Bridge CPUs (8 core, 2.6GHz) with Infiniband FDR interconnect.

As with the constant flow problem we first studied the effect of simulated faults on the error of the computed solution. The notation for the number of faults, error and wall clock times are the same as that used for the results of the constant flow computations. Table 5.5 shows results for $r = 50$ runs of the

[2]http://nci.org.au/

Table 5.6: *Numerical results for $r = 50$ samples of $u^{gcp}_{I_{n,\tau}\setminus \mathrm{supp}(U_{n,2})}$ with level $n = 22$ and truncation $\tau = 6$ (which computes 35 different $u_{\underline{i}}$). Faults were simulated with different* MTBF *(in seconds) to study the effect on error and time to solution. The simulated faults are Weibull distributed with shape $\kappa = 0.7$ in each case. Component solutions were combined 4 times throughout the computation. The computations were performed on 32 processors (4* MPI *processes with 8 OpenMP threads each). Times reported are for the inner computation loop and exclude overheads (including the load balancing and error calculation).*

MTBF	f_{ave}	ϵ_{ave}	ϵ_{min}	ϵ_{max}	w_{ave}	w_{min}	w_{max}
128	3.08	1.671e-4	1.305e-4	3.727e-4	32.99	32.56	33.43
64	4.32	1.693e-4	1.305e-4	2.938e-4	33.04	32.63	33.60
32	7.02	1.791e-4	1.305e-4	2.681e-4	33.07	32.63	33.77
16	12.02	2.191e-4	1.340e-4	5.722e-4	33.01	32.61	33.68
8	19.04	2.700e-4	1.393e-4	4.785e-4	33.22	32.64	33.79
4	31.58	3.382e-4	1.830e-4	6.305e-4	33.40	32.80	35.28

FTCT with fault simulation on a problem with $n = 22$, $\tau = 5$ consisting of 110 grids and computed on 8 MPI processes with 8 OpenMP threads each. Faults were simulated as described in Section 5.2.1 using the Weibull distribution with varying mean time between failures (MTBF) per MPI process and shape parameter 0.7. Decreasing the MTBF leads to an increase in the average number of faults that occur per run as one would expect. The time to solution is not significantly affected by the number of faults as seen in t_{ave} which varies relatively little over a 10 fold increase in the average number of faults. It is clear that the error ϵ_{ave} increases with f_{ave} and from ϵ_{min} and ϵ_{max} one has some indication that the variance in error also increases. For MTBF of 450 seconds and above the effect is relatively small. For higher frequencies of failure there is a noticeable effect but even at 27 faults per run the average error is approximately twice that without faults.

Table 5.6 again shows results for $r = 50$ runs of the FTCT with fault simulation this time with a $n = 22$, $\tau = 6$ truncated combination consisting of only 35 grids. Having less grids and therefore less redundancy this problem is more sensitive to faults. Faults were again simulated as Weibull distributed with varying means and shape parameter 0.7. The MTBF are significantly smaller here to investigate what happens to this problem for a large number of faults. The same observations can be made as in Table 5.5, namely that the time to solution is not significantly

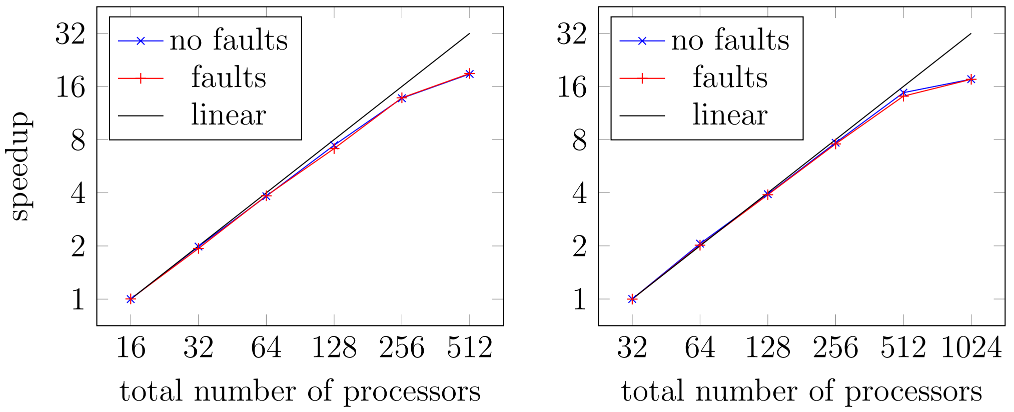


Figure 5.3: *The left plot demonstrates the parallel scaling of our implementation with a $l = 22$, $\tau = 5$ truncated combination on the left and a $l = 22$, $\tau = 4$ truncated combination on the right. Each problem has 110 and 230 coarse grids respectively. Given a fixed number of grids one observes that the scalability eventually drops off which is due to the distribution of grids to nodes. However, also notice that additional grids leads to increased parallelism. By increasing the number of grids or computing each grid over several nodes we would expect good scalability for many more processors. The* MTBF *per node used for fault simulation on the $\tau = 5$ problem was 128 seconds which led to 9 up to 36 faults occurring in each of the computations. For the $\tau = 4$ problem the* MTBF *per node was 900 seconds which led to 12 up to 41 faults occurring in each of the computations.*

affected by the failure rate and that the mean and variance of the error increases with f_{ave}. Whilst growing faster in for this problem at almost 32 faults on average the error is less than 3 times that when no faults occur.

Whilst not shown here, we have observed that if the number of grids is large enough then the expected error initially decreases as the number of faults increases. We suspect this occurs because such large numbers of grids becomes a sub-optimal use of resources and the solution error becomes dominated by terms which are additive when the combination technique is applied. As more grids fail there are less such terms in the resulting solution leading to smaller error. Eventually with enough failures the error again increases as the dominating terms become those which cancel and similar trends as above are observed from this point. Determining the number of grids which is optimal in terms of error versus time to solution is problem dependent but can be estimated by careful study of error bounds based on error splittings.

In Figure 5.3 we demonstrate the parallel scalability and efficiency of our implementation both with and without fault simulation for a reasonably high failure

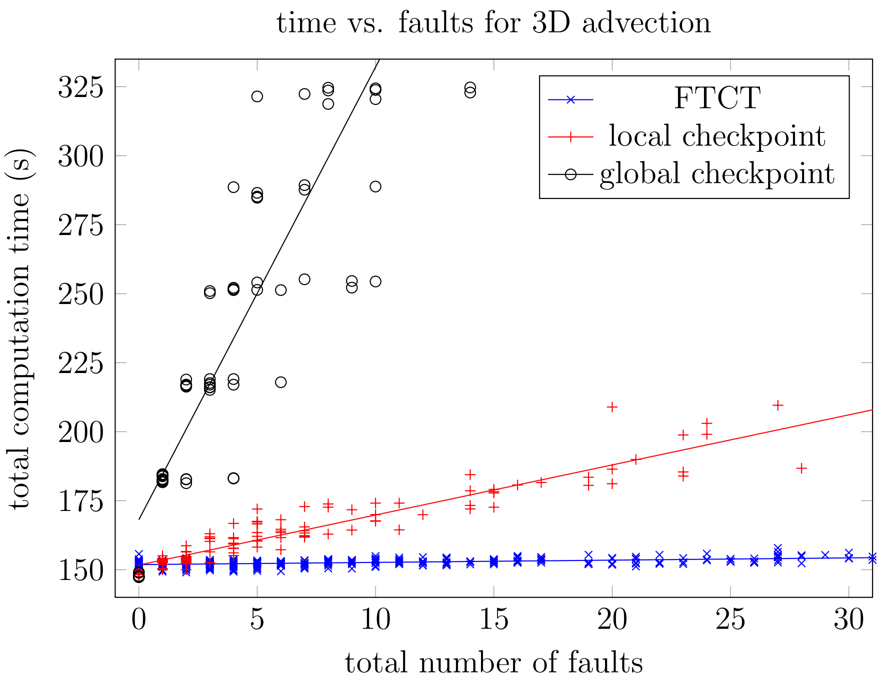


Figure 5.4: *We compare the time taken to compute the solution to the 3D advection problem using three different approaches to fault tolerance. The problem size is fixed at level* 22 *with truncation parameter* 5*. All computations used 8* MPI *processes with 8 OpenMP threads each. Component solutions are combined 4 times throughout the computation and it is during the combination that we check for faults. For each method the problem was run several times with many different* MTBF *per* MPI *process to study the effect of the number of faults on the run time.*

rate. The advection problem was solved using a $n = 22$, $\tau = 5$ truncated combination having 110 grids and a $n = 22$, $\tau = 4$ truncated combination having 230 grids. The component solutions were combined 4 times throughout each computation. The times reported here include the timing of the core of the code, that is the repeated computation, combination and communication of the solution. Start up and completion overheads including python imports, the dynamic load balancing procedure and the error calculation are excluded. It is clear that the implementation scales very well as far as the distribution of grids to nodes will allow and that the presence of faults makes no discernible difference. In particular, it is apparent that adding fault resilience has had negligible impact on the speedup of the application. Therefore, for an application that is otherwise capable of scaling to an exascale system it is not anticipated that adding fault resilience via this method would be a barrier to deployment on such a system. In our test case, by further increasing the number of grids and computing each

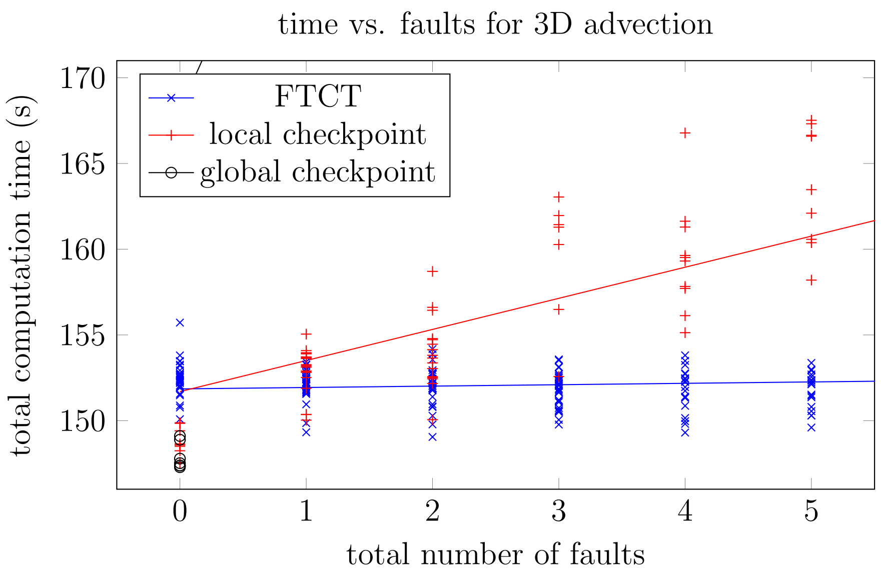


Figure 5.5: *A zoom of the bottom left corner of Figure 5.4. It is clearer that the* FTCT *approach takes slightly longer when no faults occur but with even just one fault the runs times of* FTCT *are typically better.*

grid across several nodes we expect the implementation to scale much further. We note however, that increasing the number of grids does not necessarily lead to improved error.

In Figure 5.4 we again compare the computation time required for our approach to reach a solution compared to the local and global checkpointing approach. As before, the checkpointing implementations to not require the computation of additional grids and are therefore slightly faster when no faults occur. As the number of faults increases it can be observed that the computation time for the local and global checkpointing methods begins to grow. A line of best fit makes it clear that the time for recovery with global checkpointing increases rapidly with the number of faults. Local checkpointing is a significant improvement on this but still shows growth. On the other hand our approach is barely affected by the number of faults and beats both the local and global checkpointing approaches after only a few faults.

5.3 Additional remarks and conclusions

In [67] it is discussed how a failed grid could be approximated with a sample from a grid having similar resolution. Some investigation of this approach was undertaken but it was observed that results were strongly affected by which solution grid was chosen to be sampled from. Experiments also found that this approach was not very robust, that is in some cases it provided acceptable results whilst in others the results were poor. Attempts to improve the robustness of this approach led to algorithms which typically gave the same results as the GCP. As such we felt it was unnecessary to expand upon this approach here.

Another observation to be made is that there are cases in which the GCP solution does not use all of the information from the successfully computed component solutions. For example, this often occurs when one (or more) of the finer of the component solutions (i.e. those on the top layer) which was successfully computed cannot be incorporated into the combination because of the failure of another component solution having smaller resolution. In such situations the unused component solution carries some information on the finest hierarchical surplus which can still make a contribution to the combined solution. One can simply add this hierarchical information to the combined solution. This is straightforward as the combined solution (via the GCP solution) would initially carry no information for this particular hierarchical surplus and so it may be added without affecting the rest of the combined solution.

In this thesis our fault model is largely based on hardware faults for which a process stops working. Since starting this work it has become clear that a large portion of the HPC community is interested in silent faults detection and correction. We note that the FTCT algorithm could also be used to recover from silent errors, that is where some of the data is changed without the application knowing. Of course the real challenge here is in detecting which component solutions have been affected by silent faults. It would be reasonable to expect that by comparing the approximations on different component schemes one would be able to filter out the erroneous component solution via a statistical analysis and an appropriate voting mechanism. This will require an optimisation between the number of false negatives (that is discarding component solutions which differ slightly from the others but were not affected by an error) and false positives (e.g. accepting component solutions with an error small enough not to be easily detected). Investigation into silent error detection and correction is the subject of ongoing research.

The numerical results presented in this thesis have been restricted to the scalar advection PDE. The main reason for this is that the exact solution is known and we can therefore easily compute the error and assess whether the FTCT produces results close to those obtained in the absence of faults. There has been some work at applying the FTCT to more complex problems, for example gyrokinetic plasma simulations [5, 72], and work in this direction will continue. Results thus far have been promising and give a good indication that the algorithm should work well for a large variety of problems.

Another important point to be made is that the comparisons we made to checkpoint restart routines did not use the current state of the art checkpointing. The reason for this is that checkpoint restart has developed rapidly in the last few years. Compare for example the survey article [27] and the follow up article [28]. In the former it is pointed out there is a need to research fault tolerance because existing checkpoint restart will be too costly. In the latter, an advancement of checkpoint restart methods is acknowledged that may allow such methods to be used after all. We simply did not have the resources to keep up with these advances whilst also researching a novel alternative. Future work may involve comparisons with state of the art checkpoint restart techniques with optimal checkpointing intervals.

Our results were obtained by sampling a Weibull like distribution of time between failures based on the study [116] and then simulating the effect of a fault, that is the loss of data. There has been effort to implement the FTCT with User Level Fault Mitigation (ULFM) by M. Ali such that it may one day used and tested in the presence of real hardware failures as opposed to just simulated failures.

In conclusion, we have developed a new approach to fault tolerant computation via a careful study and generalisation of the sparse grid combination technique. Additionally we have provided an extensive error analysis for this approach which is applicable to finite difference solutions of the advection equation and could be extended to a large class of hyperbolic partial differential equations. Numerical experiments based on approximate solutions to the advection equation show that the new approach is significantly better than traditional checkpointing approaches in terms of overhead and produces results that are close to that obtained in the absence of faults. Many of the results in this thesis could also be applied to many other HPC problems in which the combination technique has been applied successfully, for example uncertainty quantification, big data and inverse problems. There are several more ways in which this work may be extended going into the future. These include further error analysis for more general use cases of the FTCT algorithm, additional numerical experiments on large scale and complex problems that require significant computing resources, more detailed comparisons with alternative approaches to fault tolerance, and extensions to different nested function spaces (for example higher order sparse grid methods and nested fractal function spaces generated by iterated function systems).

Appendix A

List of Publications

The following list consists of publications I have contributed to that were submitted and/or published during the term of my PhD. Those marked with $*$ are related to the work in this thesis.

B. Harding and M. Hegland. A robust combination technique. In S. McCue, T. Moroney, D. Mallet, and J. Bunder, editors, *Proceedings of the 16th Biennial Computational Techniques and Applications Conference, CTAC-2012*, volume 54 of *ANZIAM J.*, pages C394–C411, Aug. 2013. $*$

M. Barnsley and B. Harding. Fractal transformations in 2 and 3 dimensions. *arXiv preprint arXiv:1308.6648*, August 2013.

B. Harding and M. Hegland. A parallel fault tolerant combination technique. In Bader et al. [7], pages 584–592. $*$

J. Larson, P. Strazdins, M. Hegland, B. Harding, S. Roberts, L. Stals, A. Rendell, M. Ali, and J. Southern. Managing complexity in the parallel sparse grid combination technique. In Bader et al. [7], pages 593–602. $*$

J. Larson, M. Hegland, B. Harding, S. Roberts, L. Stals, A. Rendell, P. Strazdins, M. Ali, C. Kowitz, R. Nobes, J. Southern, N. Wilson, M. Li, and Y. Oishi. Fault-tolerant grid-based solvers: Combining concepts from sparse grids and mapreduce. *Procedia Computer Science*, 18(0):130 – 139, 2013. (Proceedings of 2013 International Conference on Computational Science). $*$

M. Barnsley, B. Harding, and A. Vince. The entropy of a special overlapping dynamical system. *Ergodic Theory and Dynamical Systems*, 34(2):469–486, April 2014.

M. M. Ali, J. Southern, P. E. Strazdins, and B. Harding. Application level fault recovery: Using fault-tolerant open MPI in a PDE solver. In *2014 IEEE International Parallel & Distributed Processing Symposium Workshops, Phoenix, AZ, USA, May 19-23, 2014*, pages 1169–1178. IEEE, 2014. ∗

B. Harding and M. Hegland. Robust solutions to PDEs with multiple grids. In J. Garcke and D. Pflger, editors, *Sparse Grids and Applications - Munich 2012*, volume 97 of *Lecture Notes in Computational Science and Engineering*, pages 171–193. Springer International Publishing, 2014. ∗

M. F. Barnsley, B. Harding, and M. Rypka. Measure preserving fractal homeomorphisms. In C. Bandt, M. Barnsley, R. Devaney, K. J. Falconer, V. Kannan, and V. Kumar P.B., editors, *Fractals, Wavelets, and their Applications*, volume 92 of *Springer Proceedings in Mathematics & Statistics*, pages 79–102. Springer International Publishing, 2014.

M. Barnsley, B. Harding, A. Vince, and P. Viswanathan. Approximation of rough functions. *arXiv preprint arXiv:1412.3871*, Dec 2014.

M. F. Barnsley and B. Harding. Three-dimensional fractal homeomorphisms. in *Benoit Mandelbrot : a life in many dimensions*, New Jersey World Scientific, New Jersey, pp. 117–142, 2015.

B. Harding, M. Hegland, J. Larson, and J. Southern. Fault tolerant computation with the sparse grid combination technique. *SIAM Journal on Scientific Computing*, 37(3):C331–C353, 2015. ∗

M. M. Ali, P. E. Strazdins, B. Harding, M. Hegland, and J. W. Larson. A fault-tolerant gyrokinetic plasma application using the sparse grid combination technique. In *Proceedings of the 2015 International Conference on High Performance Computing & Simulation (HPCS 2015)*, pages 499–507, Amsterdam, The Netherlands, July 2015. Outstanding paper award. ∗

P. E. Strazdins, M. M. Ali, and B. Harding. Highly scalable algoritihms for the sparse grid combination technique. In *Proceedings of the IEEE 29th International Parallel & Distributed Processing Symposium Workshops (IPDPSW 2015)*, Hyderabad, India, May 2015. ∗

B. Harding. Adaptive sparse grids and extrapolation techniques. In J. Garcke and D. Pflüger, editors, *Sparse Grids and Applications - Stuttgart 2014*, Lecture

Notes in Computational Science and Engineering, pages 79–102. Springer International Publishing, Cham, 2016. ∗

B. Harding. Combination technique coefficients via error splittings. In J. Sharples and J. Bunder, editors, *Proceedings of the 17th Biennial Computational Techniques and Applications Conference, CTAC-2014*, volume 56 of *ANZIAM J.*, pages C355–C368, Feb. 2016. ∗

J. H. S. de Baar and B. Harding. A gradient-enhanced sparse grid algorithm for uncertainty quantification. *International Journal for Uncertainty Quantification*, 5(5):453–468, 2015. ∗

Bibliography

[1] R. Adams and J. Fournier. *Sobolev Spaces*. Pure and Applied Mathematics. Elsevier Science, 2003.

[2] V. Alexander Randall. The Eckert tapes: Computer pioneer says ENIAC team couldn't afford to fail-and didn't. *Computerworld*, 40(8), 2006.

[3] M. M. Ali, J. Southern, P. E. Strazdins, and B. Harding. Application level fault recovery: Using fault-tolerant open MPI in a PDE solver. In *2014 IEEE International Parallel & Distributed Processing Symposium Workshops, Phoenix, AZ, USA, May 19-23, 2014*, pages 1169–1178. IEEE, 2014.

[4] M. M. Ali and P. E. Strazdins. Algorithm-based master-worker model of fault tolerance in time-evolving applications. In *The Third International Conference on Performance, Safety and Robustness in Complex Systems and Applications (PESARO 2013)*, pages 40–47, Venice, Italy, April 2013.

[5] M. M. Ali, P. E. Strazdins, B. Harding, M. Hegland, and J. W. Larson. A fault-tolerant gyrokinetic plasma application using the sparse grid combination technique. In *Proceedings of the 2015 International Conference on High Performance Computing & Simulation (HPCS 2015)*, pages 499–507, Amsterdam, The Netherlands, July 2015. Outstanding paper award.

[6] L. Alvisi and K. Marzullo. Trade-offs in implementing causal message logging protocols. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, pages 58–67, New York, NY, USA, 1996. ACM.

[7] M. Bader, A. Bode, H. Bungartz, M. Gerndt, G. R. Joubert, and F. J. Peters, editors. *Parallel Computing: Accelerating Computational Science and Engineering (CSE), Proceedings of the International Conference on Parallel Computing, ParCo 2013, 10-13 September 2013, Garching (near*

Munich), Germany, volume 25 of *Advances in Parallel Computing*. IOS Press, 2014.

[8] P. Banerjee and J. Abraham. Bounds on algorithm-based fault tolerance in multiple processor systems. *Computers, IEEE Transactions on*, C-35(4):296–306, April 1986.

[9] J. Benk and D. Pflüger. Hybrid parallel solutions of the black-scholes PDE with the truncated combination technique. In W. Smari and V. Zeljkovic, editors, *2012 International Conference on High Performance Computing & Simulation, HPCS 2012, Madrid, Spain, July 2-6, 2012*, pages 678–683. IEEE, 2012.

[10] W. Bland, A. Bouteiller, T. Herault, G. Bosilca, and J. Dongarra. Post-failure recovery of MPI communication capability: Design and rationale. *International Journal of High Performance Computing Applications*, 27(3):244–254, 2013.

[11] W. Bland, A. Bouteiller, T. Herault, J. Hursey, G. Bosilca, and J. Dongarra. An evaluation of user-level failure mitigation support in MPI. *Computing*, 95(12):1171–1184, 2013.

[12] P. B. Bochev and M. D. Gunzburger. *Least-Squares Finite Element Methods*, volume 166 of *Applied Mathematical Sciences*. Springer-Verlag New York, 2009.

[13] G. Bosilca, R. Delmas, J. Dongarra, and J. Langou. Algorithm-based fault tolerance applied to high performance computing. *Journal of Parallel and Distributed Computing*, 69(4):410 – 416, 2009.

[14] A. Bouteiller, G. Bosilca, and J. Dongarra. Redesigning the message logging model for high performance. *Concurrency and Computation: Practice and Experience*, 22(16):2196–2211, 2010.

[15] A. Bouteiller, T. Herault, G. Bosilca, and J. J. Dongarra. Correlated set coordination in fault tolerant message logging protocols for many-core clusters. *Concurrency and Computation: Practice and Experience*, 25(4):572–585, 2013.

[16] P. G. Bridges, K. B. Ferreira, M. A. Heroux, and M. Hoemmen. Fault-tolerant linear solvers via selective reliability. *arXiv pre-print 1206.1390*, 2012.

[17] H. Bungartz, M. Griebel, and U. Rüde. Extrapolation, combination, and sparse grid techniques for elliptic boundary value problems. *Computer Methods in Applied Mechanics and Engineering*, 116(14):243 – 252, 1994.

[18] H.-J. Bungartz. Concepts for higher order finite elements on sparse grids. In *Houston Journal of Mathematics: Proceedings of the 3rd Int. Conf. on Spectral and High Order Methods, Houston*, pages 159–170, 1996.

[19] H.-J. Bungartz. A multigrid algorithm for higher order finite elements on sparse grids. *ETNA*, 6:6–63, 1997.

[20] H.-J. Bungartz and S. Dirnstorfer. Multivariate quadrature on adaptive sparse grids. *Computing*, 71(1):89–114, 2003.

[21] H.-J. Bungartz and S. Dirnstorfer. Higher order quadrature on sparse grids. In M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, editors, *International Conference on Computational Science*, volume 3039 of *Lecture Notes in Computer Science*, pages 394–401. Springer, 2004.

[22] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 5 2004.

[23] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise convergence of the combination technique for the Laplace equation. *East-West J. Numer. Math.*, 2:21–45, 1994. also as SFB-Bericht 342/16/93A, Institut für Informatik, TU München, 1993.

[24] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Two proofs of convergence for the combination technique for the efficient solution of sparse grid problems. In D. E. Keyes and J. Xu, editors, *Domain Decomposition Methods in Scientific and Engineering Computing, DDM7*, Contemp. Math. 180, pages 15–20. American Mathematical Society, Providence, 1994.

[25] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. A proof of convergence for the combination technique for the Laplace equation using tools of symbolic computation. *Math. Comp. in Simulation*, 42:595–605, 1996. also in G. Jacob, N. Oussous and S. Steinberg, Editoren, IMACS Symposium

on Symbolic Computation, Lille, Juni 1993. IMACS/Universite des Sciences et Technologies de Lille, Villeneuve d'Ascq, 1993 and as SFB Bericht, 342/4/93 A, Institut für Informatik, TU München, 1993.

[26] F. Cappello. Fault tolerance in petascale/ exascale systems: Current knowledge, challenges and research opportunities. *International Journal of High Performance Computing Applications*, 23(3):212–226, 2009.

[27] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir. Toward exascale resilience. *International Journal of High Performance Computing Applications*, 2009.

[28] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir. Toward exascale resilience: 2014 update. *Supercomputing frontiers and innovations*, 1(1):5–28, 2014.

[29] K. M. Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1):63–75, Feb. 1985.

[30] Z. Chen. Algorithm-based recovery for iterative methods without checkpointing. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, HPDC '11, pages 73–84, New York, NY, USA, 2011. ACM.

[31] Z. Chen and J. Dongarra. Algorithm-based fault tolerance for fail-stop failures. *Parallel and Distributed Systems, IEEE Transactions on*, 19(12):1628–1641, Dec 2008.

[32] P. Ciarlet. *Linear and Nonlinear Functional Analysis with Applications:*. Applied mathematics. Society for Industrial and Applied Mathematics, 2013.

[33] C. Constantinescu. Trends and challenges in VLSI circuit reliability. *Micro, IEEE*, 23(4):14–19, July 2003.

[34] R. Courant, K. Friedrichs, and H. Lewy. ber die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.

[35] D. Cox. *Renewal Theory*. Methuen's monographs on applied probability and statistics. Methuen, 1962.

[36] B. Davey and H. Priestley. *Introduction to Lattices and Order*. Cambridge mathematical text books. Cambridge University Press, 2002.

[37] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig, F. Cappello, B. Chapman, X. Chi, A. Choudhary, S. Dosanjh, T. Dunning, S. Fiore, A. Geist, B. Gropp, R. Harrison, M. Hereld, M. Heroux, A. Hoisie, K. Hotta, Z. Jin, Y. Ishikawa, F. Johnson, S. Kale, R. Kenway, D. Keyes, B. Kramer, J. Labarta, A. Lichnewsky, T. Lippert, B. Lucas, B. Maccabe, S. Matsuoka, P. Messina, P. Michielse, B. Mohr, M. S. Mueller, W. E. Nagel, H. Nakashima, M. E. Papka, D. Reed, M. Sato, E. Seidel, J. Shalf, D. Skinner, M. Snir, T. Sterling, R. Stevens, F. Streitz, B. Sugar, S. Sumimoto, W. Tang, J. Taylor, R. Thakur, A. Trefethen, M. Valero, A. Van Der Steen, J. Vetter, P. Williams, R. Wisniewski, and K. Yelick. The international exascale software project roadmap. *Int. J. High Perform. Comput. Appl.*, 25(1):3–60, Feb. 2011.

[38] P. Du, A. Bouteiller, G. Bosilca, T. Herault, and J. Dongarra. Algorithm-based fault tolerance for dense matrix factorizations. In *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '12, pages 225–234, New York, NY, USA, 2012. ACM.

[39] D. Duffy. *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*. The Wiley Finance Series. Wiley, 2006.

[40] E. Elnozahy and J. Plank. Checkpointing for peta-scale systems: a look into the future of practical rollback-recovery. *Dependable and Secure Computing, IEEE Transactions on*, 1(2):97–108, April 2004.

[41] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Comput. Surv.*, 34(3):375–408, Sept. 2002.

[42] C. Engelmann and A. Geist. A diskless checkpointing algorithm for super-scale architectures applied to the fast Fourier transform. *Challenges of Large Applications in Distributed Environments, International Workshop on*, 0:47, 2003.

[43] G. Fagg and J. Dongarra. FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world. In J. Dongarra, P. Kacsuk, and N. Podhorszki, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 1908 of *Lecture Notes in Computer Science*, pages 346–353. Springer Berlin Heidelberg, 2000.

[44] G. E. Fagg, A. Bukovsky, and J. J. Dongarra. HARNESS and fault tolerant MPI. *Parallel Computing*, 27(11):1479 – 1495, 2001. Clusters and computational grids for scientific computing.

[45] G. E. Fagg and J. J. Dongarra. Building and using a fault-tolerant MPI implementation. *International Journal of High Performance Computing Applications*, 18(3):353–361, 2004.

[46] G. E. Fagg, E. Gabriel, G. Bosilca, T. Angskun, Z. Chen, J. Pjesivac-grbovic, K. London, and J. J. Dongarra. Extending the MPI specification for process fault tolerance on high performance computing systems. In *In Proceeding of International Supercomputer Conference (ICS*, 2003.

[47] G. E. Fagg, E. Gabriel, Z. Chen, T. Angskun, G. Bosilca, J. Pjesivac-Grbovic, and J. J. Dongarra. Process fault tolerance: Semantics, design and applications for high performance computing. *International Journal of High Performance Computing Applications*, 19(4):465–477, 2005.

[48] W. Feller. *An introduction to probability theory and its applications*. Wiley series in probability and mathematical statistics: Probability and mathematical statistics. Wiley, 1971.

[49] A. Gainaru, F. Cappello, M. Snir, and W. Kramer. Fault prediction under the microscope: A closer look into hpc systems. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pages 1–11, Nov 2012.

[50] J. Garcke. Sparse grids in a nutshell. In J. Garcke and M. Griebel, editors, *Sparse Grids and Applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 57–80. Springer Berlin Heidelberg, 2013.

[51] C. Gauss. *Carl Friedrich Gauss Werke: Neunter band [IX]*. Königlichen Gesellschaft der Wissenschaften, 1903.

[52] A. Geist and C. Engelmann. Development of naturally fault tolerant algorithms for computing on 100,000 processors. *Journal of Parallel and Distributed Computing*, 2002.

[53] S. K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Mat. Sb. (N.S.)*, 47 (89):271–306, 1959.

[54] T. Görler. *Multiscale Effects in Plasma Microturbulence.* PhD thesis, Universität Ulm, 2009.

[55] J. Gray. A census of tandem system availability between 1985 and 1990. *Reliability, IEEE Transactions on*, 39(4):409–418, Oct 1990.

[56] M. Griebel. Parallel multigrid methods on sparse grids. In *Multigrid methods III*, ISNM 98, pages 211–221. Birkhäuser, Basel, 1991.

[57] M. Griebel. Adaptive sparse grid multilevel methods for elliptic pdes based on finite differences. *Computing*, 61(2):151–179, 1998.

[58] M. Griebel and H. Harbrecht. On the convergence of the combination technique. In J. Garcke and D. Pflger, editors, *Sparse Grids and Applications - Munich 2012*, volume 97 of *Lecture Notes in Computational Science and Engineering*, pages 55–74. Springer International Publishing, 2014.

[59] M. Griebel, W. Huber, U. Rüde, and T. Strtkuhl. The combination technique for parallel sparse-grid-preconditioning or -solution of pde's on workstation networks. In L. Boug, M. Cosnard, Y. Robert, and D. Trystram, editors, *Parallel Processing: CONPAR 92VAPP V*, volume 634 of *Lecture Notes in Computer Science*, pages 217–228. Springer Berlin Heidelberg, 1992.

[60] M. Griebel and T. Schiekofer. An adaptive sparse grid Navier-Stokes solver in 3D based on finite differences. In H. Bock, G. Kanschat, R. Rannacher, F. Brezzi, R. Glowinski, Y. Kuznetsov, and J. Periaux, editors, *Proceedings ENUMATH97*, Heidelberg, 1999. World Scientific Publishing.

[61] M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens,

editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992. also as SFB Bericht, 342/19/90 A, Institut für Informatik, TU München, 1990.

[62] M. Griebel, C. Zenger, and S. Zimmer. Multilevel Gauß–Seidel–algorithms for full and sparse grid problems. *Computing*, 50:127–148, 1993.

[63] M. Griebel and G. W. Zumbusch. Adaptive sparse grids for hyperbolic conservation laws. In M. Fey and R. Jeltsch, editors, *Hyperbolic Problems: Theory, Numerics, Applications. 7th International Conference in Zürich, February 1998*, volume 1 of *International Series of Numerical Mathematics 129*, pages 411–422, Basel, Switzerland, 1999. Birkhäuser.

[64] B. Harding. Adaptive sparse grids and extrapolation techniques. In J. Garcke and D. Pflüger, editors, *Sparse Grids and Applications - Stuttgart 2014*, Lecture Notes in Computational Science and Engineering, pages 79–102. Springer International Publishing, Cham, 2016.

[65] B. Harding. Combination technique coefficients via error splittings. In J. Sharples and J. Bunder, editors, *Proceedings of the 17th Biennial Computational Techniques and Applications Conference, CTAC-2014*, volume 56 of *ANZIAM J.*, pages C355–C368, Feb. 2016.

[66] B. Harding and M. Hegland. A parallel fault tolerant combination technique. In Bader et al. [7], pages 584–592.

[67] B. Harding and M. Hegland. A robust combination technique. In S. McCue, T. Moroney, D. Mallet, and J. Bunder, editors, *Proceedings of the 16th Biennial Computational Techniques and Applications Conference, CTAC-2012*, volume 54 of *ANZIAM J.*, pages C394–C411, Aug. 2013.

[68] B. Harding and M. Hegland. Robust solutions to PDEs with multiple grids. In J. Garcke and D. Pflger, editors, *Sparse Grids and Applications - Munich 2012*, volume 97 of *Lecture Notes in Computational Science and Engineering*, pages 171–193. Springer International Publishing, 2014.

[69] B. Harding, M. Hegland, J. Larson, and J. Southern. Fault tolerant computation with the sparse grid combination technique. *SIAM Journal on Scientific Computing*, 37(3):C331–C353, 2015.

[70] M. Hegland. Adaptive sparse grids. In K. Burrage and R. B. Sidje, editors, *Proc. of 10th Computational Techniques and Applications Conference CTAC-2001*, volume 44, pages C335–C353, Apr. 2003. [Online] `http://anziamj.austms.org.au/V44/CTAC2001/Hegl` [April 1, 2003].

[71] M. Hegland, J. Garcke, and V. Challis. The combination technique and some generalisations. *Linear Algebra and its Applications*, 420(23):249 – 275, 2007.

[72] A. P. Hinojosa, C. Kowitz, M. Heene, D. Pflüger, and H.-J. Bungartz. Towards a fault-tolerant, scalable implementation of GENE. In M. Mehl, M. Bischoff, and M. Schäfer, editors, *Recent Trends in Computational Engineering - CE2014: Optimization, Uncertainty, Parallel Algorithms, Coupled and Complex Problems*, pages 47–65, Cham, 2015. Springer International Publishing.

[73] R. Hochmuth, S. Knapek, and G. Zumbusch. Tensor products of Sobolev spaces and applications. Technical Report 685, SFB 256, Univ. Bonn, 2000.

[74] K.-H. Huang and J. Abraham. Algorithm-based fault tolerance for matrix operations. *Computers, IEEE Transactions on*, C-33(6):518–528, June 1984.

[75] W. Hundsdorfer and J. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations.* Springer Series in Computational Mathematics. Springer, 2003.

[76] S. Karlin and H. Taylor. *A First Course in Stochastic Processes.* Elsevier Science, 1975.

[77] R. Karp. Reducibility among combinatorial problems. In R. Miller, J. Thatcher, and J. Bohlinger, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US, 1972.

[78] D. I. Ketcheson, S. Gottlieb, and C. B. Macdonald. Strong stability preserving two-step Runge-Kutta methods. *SIAM Journal on Numerical Analysis*, 49(6):2618–2639, 2011.

[79] J. Larson, M. Hegland, B. Harding, S. Roberts, L. Stals, A. Rendell, P. Strazdins, M. Ali, C. Kowitz, R. Nobes, J. Southern, N. Wilson, M. Li,

and Y. Oishi. Fault-tolerant grid-based solvers: Combining concepts from sparse grids and mapreduce. *Procedia Computer Science*, 18(0):130 – 139, 2013. (Proceedings of 2013 International Conference on Computational Science).

[80] J. W. Larson, P. E. Strazdins, M. Hegland, B. Harding, S. G. Roberts, L. Stals, A. P. Rendell, M. M. Ali, and J. Southern. Managing complexity in the parallel sparse grid combination technique. In Bader et al. [7], pages 593–602.

[81] B. Lastdrager, B. Koren, and J. Verwer. Solution of time-dependent advection-diffusion problems with the sparse-grid combination technique and a Rosenbrock solver. *Computational Methods in Applied Mathematics*, 1(1):86–99, 2001.

[82] B. Lastdrager, B. Koren, and J. Verwer. The sparse-grid combination technique applied to time-dependent advection problems. *Applied Numerical Mathematics*, 38(4):377 – 401, 2001.

[83] P. Lax and B. Wendroff. Systems of conservation laws. *Communications on Pure and Applied Mathematics*, 13(2):217–237, 1960.

[84] P. D. Lax and R. D. Richtmyer. Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics*, 9(2):267–293, 1956.

[85] J. Lee and B. Fornberg. A split step approach for the 3-D Maxwell's equations. *Journal of Computational and Applied Mathematics*, 158(2):485 – 505, 2003.

[86] C. Leentvaar. *Pricing multi-asset options with sparse grids.* PhD thesis, Technische Universiteit Delft, 2008.

[87] P. Lemarinier, A. Bouteiller, T. Herault, G. Krawezik, and F. Cappello. Improved message logging versus improved coordinated checkpointing for fault tolerant MPI. In *Cluster Computing, 2004 IEEE International Conference on*, pages 115–124, Sept 2004.

[88] R. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations.* Society for Industrial and Applied Mathematics, 2007.

[89] C. Liem, T. Lü, and T. Shih. *The Splitting Extrapolation Method: A New Technique in Numerical Solution of Multidimensional Problems.* Series on applied mathematics. World Scientific, 1995.

[90] H. Liu. An algorithm-based recovery scheme for extreme scale computing. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 2010–2013, May 2011.

[91] Y. Liu, R. Nassar, C. Leangsuksun, N. Naksinehaboon, M. Pun, and S. Scott. An optimal checkpoint/restart model for a large scale high performance computing system. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–9, April 2008.

[92] H. Ltaief, E. Gabriel, and M. Garbey. Fault tolerant algorithms for heat transfer problems. *Journal of Parallel and Distributed Computing*, 68(5):663 – 677, 2008.

[93] C.-D. Lu. *Scalable diskless checkpointing for large parallel systems.* PhD thesis, University of Illinois at Urbana-Champaign, 2005.

[94] F. T. Luk and H. Park. An analysis of algorithm-based fault tolerance techniques. *J. Parallel Distrib. Comput.*, 5(2):172–184, Apr. 1988.

[95] S. M. R. Mark Brown. Asymptotic properties of cumulative processes. *SIAM Journal on Applied Mathematics*, 22(1):93–105, 1972.

[96] F. Mattern. Efficient algorithms for distributed snapshots and global virtual time approximation. *Journal of Parallel and Distributed Computing*, 18(4):423 – 434, 1993.

[97] E. M. McElwee. Statistical evaluation of life expectancy of vacuum tubes designed for long-life operation. *Proceedings of the IRE*, 39(2):137–141, Feb 1951.

[98] R. I. McLachlan and G. R. W. Quispel. Splitting methods. *Acta Numerica*, 11:341–434, 1 2002.

[99] J. Medhi. *Stochastic Processes.* J. Wiley, 1994.

[100] G. E. Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *Solid-State Circuits Society Newsletter, IEEE*, 11(5):33–35, Sept 2006.

[101] V. Nelson. Fault-tolerant computing: fundamental concepts. *Computer*, 23(7):19–25, July 1990.

[102] A. Oliner and J. Stearley. What supercomputers say: A study of five system logs. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 575–584, June 2007.

[103] S. Pauli, M. Kohler, and P. Arbenz. A fault tolerant implementation of multi-level Monte Carlo methods. In Bader et al. [7], pages 471–480.

[104] C. Pflaum and A. Zhou. Error analysis of the combination technique. *Numerische Mathematik*, 84(2):327–350, 1999.

[105] I. Philp. Software failures and the road to a petaflop machine. In *HPCRI: 1st Workshop on High Performance Computing Reliability Issues, in Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA-11)*, 2005.

[106] J. Plank and K. Li. Faster checkpointing with n+1 parity. In *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, pages 288–297, June 1994.

[107] J. Plank, K. Li, and M. Puening. Diskless checkpointing. *Parallel and Distributed Systems, IEEE Transactions on*, 9(10):972–986, Oct 1998.

[108] J. S. Plank, Y. Kim, and J. J. Dongarra. Fault-tolerant matrix operations for networks of workstations using diskless checkpointing. *Journal of Parallel and Distributed Computing*, 43(2):125 – 138, 1997.

[109] C. Reisinger. *Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben.* PhD thesis, Universität Heidelberg, 2004.

[110] C. Reisinger. Analysis of linear difference schemes in the sparse grid combination technique. *IMA Journal of Numerical Analysis*, 2012.

[111] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 210(459-470):307–357, 1911.

[112] R. Richtmyer and K. Morton. *Difference Methods for Initial-Value Problems*, volume 4 of *Interscience tracts in pure and applied mathematics*. Interscience Publishers, second edition, 1967.

[113] S. Ross. *Introduction to Probability Models*. Elsevier Science, 2014.

[114] A. Roy-Chowdhury and P. Banerjee. Algorithm-based fault location and recovery for matrix computations. In *Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on*, pages 38–47, June 1994.

[115] U. Rüde. The hierarchical basis extrapolation method. *SIAM Journal on Scientific and Statistical Computing*, 13(1):307–318, 1992.

[116] B. Schroeder and G. Gibson. A large-scale study of failures in high-performance computing systems. *Dependable and Secure Computing, IEEE Transactions on*, 7(4):337–350, Oct 2010.

[117] B. Schroeder and G. A. Gibson. Understanding failures in petascale computers. *Journal of Physics: Conference Series*, 78(1):012022, 2007.

[118] J. Sloan, D. Kesler, R. Kumar, and A. Rahimi. A numerical optimization-based methodology for application robustification: Transforming applications for error tolerance. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 161–170, June 2010.

[119] S. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Mathematics, Doklady*, 4:240–243, 1963.

[120] P. E. Strazdins, M. M. Ali, and B. Harding. Highly scalable algoritihms for the sparse grid combination technique. In *Proceedings of the IEEE 29th International Parallel & Distributed Processing Symposium Workshops (IPDPSW 2015)*, Hyderabad, India, May 2015.

[121] K. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. PHI Learning, 1988.

[122] E. Turkel. Symmetric hyperbolic difference schemes and matrix problems. *Linear Algebra and its Applications*, 16(2):109 – 129, 1977.

[123] J. VonNeumann and R. D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21(3):232–237, 1950.

[124] R. Wang, E. Yao, M. Chen, G. Tan, P. Balaji, and D. Buntinas. Building algorithmically nonstop fault tolerant MPI programs. In *High Performance Computing (HiPC), 2011 18th International Conference on*, pages 1–9, Dec 2011.

[125] W. Whitt. Approximating a point process by a renewal process, I: Two basic methods. *Operations Research*, 30(1):125–147, 1982.

[126] R. Wolff. *Stochastic Modeling and the Theory of Queues.* Prentice-Hall international series in industrial and systems engineering. Prentice Hall, 1989.

[127] M. Wong. *Theory of the sparse grid combination technique.* PhD thesis, Australian National University, 2014.

[128] E. Yao, R. Wang, M. Chen, G. Tan, and N. Sun. A case study of designing efficient algorithm-based fault tolerant application for exascale parallelism. In *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 438–448, May 2012.

[129] H. Yoshida. Construction of higher order symplectic integrators. *Physics Letters A*, 150(57):262 – 268, 1990.

[130] J. W. Young. A first order approximation to the optimum checkpoint interval. *Commun. ACM*, 17(9):530–531, Sept. 1974.

[131] C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proc. Sixth GAMM-Seminar, Kiel*, pages 191–211. Vieweg-Verlag (1991), 1990.