DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Evaluation of different time-synchronization schemes for the combination technique

Thomas Bellebaum

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Evaluation of different time-synchronization schemes for the combination technique

Evaluation verschiedener Zeitsynchronisationsschemata für die Kombinationstechnik

Author:	Thomas Bellebaum
Supervisor:	Prof. Dr. rer. nat. habil. Hans-Joachim Bungartz
Advisor:	M.Sc. Michael Obersteiner
Submission Date:	September 15th, 2018

I confirm that this bachelor’s thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, September 15th, 2018 Thomas Bellebaum

Abstract

Solving partial differential equations (PDEs) numerically via discretization on regular grids in high dimensions is computationally demanding. The sparse grid combination technique is known to reduce the needed work for certain problems by combining several smaller grids with different levels. Due to the CFL-condition, to allow for convergence using an explicit solving method, timesteps may only be chosen based on the spacial mesh widths of the grids. This thesis aims to exploit the CFL-condition on every individual component grid to maximize taken timesteps and achieve a minimal asymtotic computational complexity for the integration. Theoretical results are tested using sample PDEs for complexity and accuracy.

Zusammenfassung

Das numerische Lösen partieller Differentialgleichungen mittels Diskretisierung auf reguläre Gitter ist sehr anspruchsvoll mit Blick auf benötigte Resourcen. Die Kombinationstechnik für Sparse Grids kann die nötige Arbeit für einige Probleme durch Kombination mehrerer kleinerer Gitter mit verschiedenen Leveln deutlich reduzieren. Aufgrund der CFL-Bedingung können nur Zeitschritte zur Integration gewählt werden, welche bestimmte Bedingungen in Abhängigkeit der Maschenweite der Gitter erfüllen. Diese Bachelorarbeit beschäftigt sich damit, Zeitschritte anhand der Limitationen der CFL-Bedingung zu maximieren und den asymptotischen rechnerischen Aufwand für eine Integration zu minimieren. Theoretische Ergebnisse werden getestet mittels beispielhafter partieller Differentialgleichungen auf rechnerischen Aufwand und Genauigkeit der Lösungen.

Contents

1 Introduction

Equations describing nature, as they are developed in almost every natural science, such as physics, chemistry or biology, often model the interdependency between several quantities like electric current and magnetic field strength. Such equations usually describe systems not only statically, but how they evolve over time. Not only is it of interest how large quantities at certain points in space are, but also the rate at which they are changing as time progresses and as one moves away from that particular point. Equations capable of incorporating above ideas are partial differential equations (PDEs).

There are several approaches to solving PDEs. Certain branches of mathematics are devoted to finding exact solutions to the development of those systems. However, finding exact solutions to complicated PDEs is not always feasible. E.g. the Navier-Stokes equations are capable of describing fluid flow in three dimensions, yet the very existence of a solution to those equations has to this date not been proven. But for practical applications, like creating smoke for a movie effect, approximate solutions created using various numerical methods are sufficient.

Many numerical approaches require the discretization of an interval of a function onto a regular grid. While the approximation quality generally increases as the mesh width decreases for functions which fulfill certain smoothness conditions, the amount of grid points increases. In higher dimensional problems this rapid increase prevents an efficient, accurate solving of the PDEs, a problem known as the curse of dimensionality. Sparse grids are known to counter this increase and reduce the number of grid points while keeping the approximation quality high. Nevertheless, working with them requires highly specialized algorithms. This is where the combination technique comes in. It combines discrete functions on grids with lower approximation quality (called component grids) to a function for a sparse grid.

If one discretizes time and uses a grid to describe the systems state at any discrete point in time, and any new grid is to be extracted from the old ones, certain criteria involving the timestep between two grids and the arrangement of grid points within the grids have to be met in order to allow quantities (like water waves) to move across space as time progresses. Using the combination technique, these criteria allow for different timesteps on the component grids. In the following thesis, I evaluate the utility of maximizing the timesteps on every component grid, thereby reducing the total number of grid point values to be calculated within the combination technique.

This thesis is structured as follows: In chapter 2 basic concepts are formally introduced including the combination technique and the CFL-condition. Chapter 3 consists of a theoretical evaluation of different time stepping strategies, thereby revealing their asymptotic computational complexity. In chapter 4 the results from chapter 3 will be tested as well as the benefits of different time synchronization strategies and dynamic timestep adjustments. And lastly chapter 5 will give a concluding overview of the matter.

2 Basics

2.1 Finite Difference Method

A common approach to solving partial differential equations $f(t, \overrightarrow{x}, u, \frac{\delta u}{\delta t}, ...) = 0$ with known initial conditions is the so-called finite difference method (FDM). For this method the initial function gets discretized on a grid. The derivatives in the PDE then get approximated by finite differences. Thereby the differential equation gets transformed into a difference equation. One hopes that the solution to the difference equation will converge towards the solution of the differential equation as the mesh width of the underlying grid approaches zero. However, as we will see, there are some necessity conditions for convergence. In the following I will assume a regular grid with d dimensions in space and one dimension in time.

2.1.1 First Order Derivatives

Approximating the functions' derivative locally at some point $\overrightarrow{x}$ by a linear function through points on that function with offsets a and b, having a distance of $h := b - a$, we get

$$\forall i \frac{\partial f(\overrightarrow{x})}{\partial x_i} \approx \frac{f(\overrightarrow{x} + b * \overrightarrow{e_i}) - f(\overrightarrow{x} + a * \overrightarrow{e_i})}{h}$$

where e_i represents the i-th unit vector. h will naturally often be a multiple of the mesh width of the underlying regular grid. Common choices for a and b are

- the forward difference: $\frac{\partial f(\overrightarrow{x})}{\partial x_i} \approx \frac{f(\overrightarrow{x}+h*\overrightarrow{e_i})-f(\overrightarrow{x})}{h}$
- the backward difference: $\frac{\partial f(\overrightarrow{x})}{\partial x_i} \approx \frac{f(\overrightarrow{x})-f(\overrightarrow{x}-h*\overrightarrow{e_i})}{h}$
- the central difference: $\frac{\partial f(\overrightarrow{x})}{\partial x_i} \approx \frac{f(\overrightarrow{x}+h*\overrightarrow{e_i})-f(\overrightarrow{x}-h*\overrightarrow{e_i})}{2h}$

For the forward and backward differences, the error is in $\mathcal{O}(h)$, while for the central difference, it is in $\mathcal{O}(h^2)$. If one seeks for higher order convergence of the derivative approximation, it is also possible to approximate the function locally by a higher order polynome by taking into account more than two gridpoints.

2.1.2 Higher Order Derivatives

Approximating derivatives of higher order is straight foreward, since $\frac{\partial^2 f(\vec{x})}{\partial x_i^2} = \frac{\partial}{\partial x_i}(\frac{\partial}{\partial x_i} f(\vec{x}))$. For example, using central differences, we have $\frac{\partial^2 f(\vec{x})}{\partial x_i^2} \approx \frac{\frac{f(\vec{x}+h*\vec{e_i})-f(\vec{x})}{h} - \frac{f(\vec{x})-f(\vec{x}-h*\vec{e_i})}{h}}{h} = \frac{f(\vec{x}+h*\vec{e_i})-2f(\vec{x})+f(\vec{x}-h*\vec{e_i})}{h^2}$. Again, an increase in the approximation-polynome order can increase the order of convergence for the approximation. This principle extends into derivatives in multiple variables.

2.1.3 Explicit solving method

When a PDE $f(t, \vec{x}, u, \frac{\delta u}{\delta t}, ...) = 0$ is transformed into a difference equation, it may be rearranged to reveal a formula to calculate a function value at a grid point at the next timestep using an explicit Euler method.

I will demonstrate this method via the 2-dimensional advection equation

$$\frac{\partial f}{\partial t} + a\frac{\partial f}{\partial x} + b\frac{\partial f}{\partial y} = 0$$

Using the forward difference in time and central difference in space, we get

$$\frac{f^{n+1}(x,y) - f^n(x,y)}{h_t} + a * \frac{f^n(x+h_x,y) - f^n(x-h_x,y)}{2*h_x} + b * \frac{f^n(x,y+h_y) - f^n(x,y-h_y)}{2*h_y} = 0$$

When solving for $f^{n+1}(x,y)$ we see that

$$f^{n+1}(x,y) = f^n(x,y) - h_t * \left(a * \frac{f^n(x+h_x,y) - f^n(x-h_x,y)}{2*h_x} + b * \frac{f^n(x,y+h_y) - f^n(x,y-h_y)}{2*h_y}\right)$$

This formula will only work for non-boundary grid points. For boundary points it is common to either have them fixed according to a formula, or derive them using a different method for approximating the derivatives (e.g. on the lower end of the x coordinate, forward difference may be used instead of central difference). Alternativaly, one may loop the grid.

2.2 Single Grids

In the following I will limit myself (without loss of generality) to grids representing the domain $[0,1]^d$. I will adapt a notation close to [Gar13].

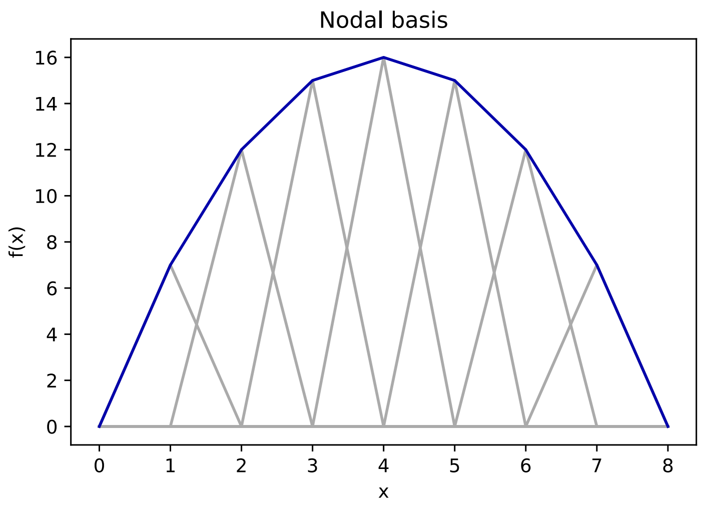


Figure 2.1: Approximation of $f(x) = -x^2 + 8x$ on a one-dimensional grid. The corresponding hat functions are depicted in grey.

Let $\vec{l} \in \mathbb{N}_0^d$ denote the level-vector. We define a grid with regular mesh width of $h_{x_i} := 2^{-l_i}$ for $i \in [d]$ in the i-th cardinal direction via it's points

$$\vec{x}_{\vec{l},\vec{j}} := (x_{l_1,j_1}, ..., x_{l_d,j_d})$$

having $x_{l_i,j_i} := j_i \cdot h_{x_i}$ with $j_i \in \mathbb{Z}_{2^{l_i}+1}$.

A grid can represent functions in

$$V_{\vec{l}} := span\{\Phi_{\vec{l},\vec{j}} \mid j_t \in \mathbb{Z}_{2^{l_t}+1},\ t \in [d]\}$$

where

$$\Phi_{\vec{l},\vec{j}}(\vec{x}) := \Pi_{t=1}^d \Phi_{l_t,j_t}(x_t)$$

and Φ_{l_t,j_t} is a one-dimensional hat-function, defined by:

$$\Phi_{l_t,j_t}(x) = max\{0, 1 - |x_t/h_{x_t} - j_t|\}$$

An examplary function is depicted in figure 2.1.

In leading order a single grid has $\mathcal{O}(\Pi_{t=1}^d 2^{l_t}) = \mathcal{O}(2^{|\vec{l}|_1})$ grid points. When we assume a level-vector $\vec{l_n}$ given by $(l_n)_t := n$ in every kardinal direction t, i.e. increasing n by 1 is the equivalent to halving the mesh width in every direction, we see that the number of grid points is given by $\mathcal{O}(2^{nd})$.

N.B. the number of points is normally displayed according to a mesh width $h := 2^{-n}$ relative to the mesh width of $\vec{l_0}$. The number of points in a single regular grid is then written as $\mathcal{O}(h^{-d})$. This exponential correlation of the dimension d and the number of grid points is known as the curse of dimensionality.

2.3 Sparse Grids

To deal with the curse of dimensionality, sparse grids are known to reduce the number of grid points while keeping the increase in the discretization error low. They were formalized in [Smo63]. I will however continue to use a notation close to [Gar13].

To be able to remove some gridpoints without causing a massive error (using the current definition a removal would mean that the function is always 0 at that point), one first has to reformulate it in the hierarchical basis.

To do this, the function space $V_{\vec{l}}$ then gets partitioned into the so called hierarchical difference spaces

$$W_{\vec{l}} := V_{\vec{l}} \setminus \bigoplus_{t\in[d]\wedge l_t>0} V_{\vec{l}-\vec{e_t}}$$

where $\vec{e_t}$ is the t-th unit vector. It holds that

$$V_{\vec{l}} = \bigoplus_{\vec{0}\leq\vec{k}\leq\vec{l}} W_{\vec{k}}$$

where I define $\vec{k} \leq \vec{l}$ as $k_t \leq l_t \forall t \in [d]$. Especially, defining $V_n := V_{\vec{n}}$, the formula can be rewritten as

$$V_n = \bigoplus_{\vec{0}\leq\vec{k}\wedge|\vec{k}|_\infty\leq n} W_{\vec{k}}$$

Notice, however, that this reformulation of $V_{\vec{l}}$ has lead to a representation using hat functions of different sizes, since, defining

$$B_{\vec{l}} := \{\vec{j} \in \mathbb{N}^d | (l_t > 0 \to j_t \in [2^{l_t}] \setminus 2\mathbb{N}) \wedge (l_t = 0 \to j_t \in \{0,1\})\}$$

we see that

$$W_{\vec{l}} = span\{\Phi_{\vec{l},\vec{j}} | \vec{j} \in B_{\vec{l}}\}$$

and the domain volume of $\Phi_{\vec{l},\vec{j}}$ depends on $\vec{l}$.

Every function can now uniquely be represented on our grid as

$$f(\vec{x}) = \sum_{\vec{0}\leq\vec{l}\wedge|\vec{l}|_\infty\leq n} \sum_{\vec{j}\in B_{\vec{l}}} \alpha_{\vec{l},\vec{j}} \cdot \Phi_{\vec{l},\vec{j}}(\vec{x})$$

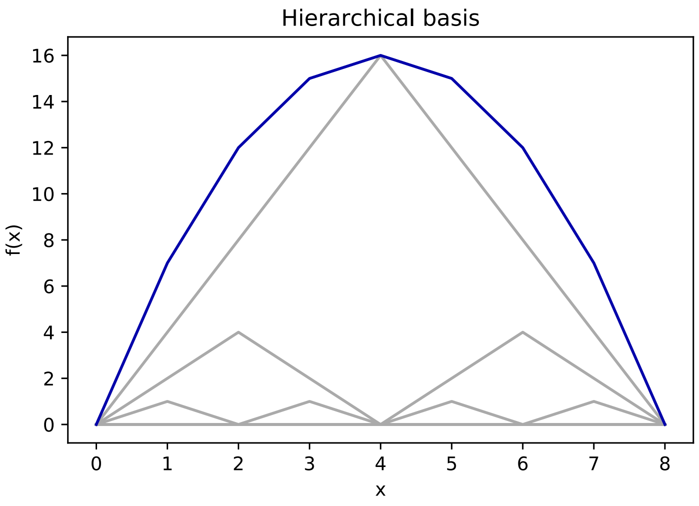


Figure 2.2: Approximation of $f(x) = -x^2 + 8x$ on a one-dimensional grid using hat functions of different support sizes.

where the coefficient α is in one dimension given by

$$\alpha l, j = f(x_{l,j}) - \frac{f(x_{l,j} - h_x) + f(x_{l,j} + h_x)}{2}$$

which can be written as the operator

$$\alpha_{l,j} = [-\frac{1}{2} \quad 1 \quad -\frac{1}{2}]_{l,j} f$$

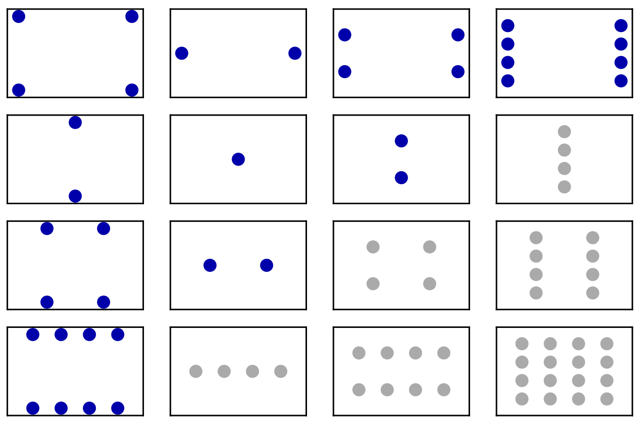
and in d dimensions by

$$\alpha_{\vec{l},\vec{j}} = \left(\Pi_{t=1}^{d}[-\frac{1}{2} \quad 1 \quad -\frac{1}{2}]_{l_t,j_t}\right) f.$$

It shall be noted that for boundary values in the one-dimensional case we define $\alpha_{0,j} = f(x_{0,x})$. Figure 2.2 depicts a function in the new basis.

Leaving out hierarchical difference spaces that hold many grid points, but do not decrease the error by much, i.e. $W_{\vec{k}}$ with $|\vec{k}|_1 > n$ for some n, one retrieves the sparse grid function space V_n^s as

$$V_n^s = \bigoplus_{\vec{0} \leq \vec{k} \wedge |\vec{k}|_1 \leq n} W_{\vec{k}}$$



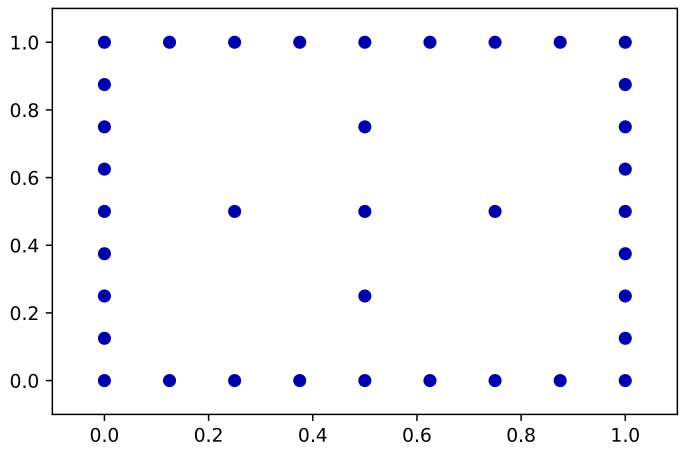


Figure 2.3: 2-Dimensional hierarchical difference spaces for up to level $(3, 3)$. Difference spaces in blue are considered for the depicted sparse grid with $n = 3$

We will now take a look at the number of grid points in a sparse grid. Any calculations in this section follow [Gar13] with slight adjustments to the notation. According to the last formula, we have

$$|V_n^s| = |\bigoplus_{\vec{0} \leq \vec{k} \wedge |\vec{k}|_1 \leq n} W_{\vec{k}}|,$$

which, according to the definition of $B_{\vec{l}}$, incorporates

$$\sum_{\vec{0} \leq \vec{l} \wedge |\vec{l}|_1 \leq n} 2^{|\vec{l} - \vec{1}|_1} = 2^{-d} \sum_{i=0}^{n} 2^i * \sum_{\vec{0} \leq \vec{l} \wedge |\vec{l}|_1 \leq i} 1 = 2^{-d} \sum_{i=0}^{n} 2^i * \binom{i+d-1}{d-1}$$

grid points.

We will represent the sum as the $(d-1)$-th derivative of a function evaluated at $x = 2$:

$$\begin{aligned}
&\sum_{i=0}^{n} x^i * \binom{i+d-1}{d-1} \\
&= \frac{1}{(d-1)!} \sum_{j=0}^{n} (x^{i+d-1})^{(d-1)} \\
&= \frac{1}{(d-1)!} (x^{d-1} * \frac{1-x^{n+1}}{1-x})^{(d-1)} \\
&= \frac{1}{(d-1)!} \sum_{k=0}^{d-1} \binom{d-1}{k} (x^{d-1} - x^{n+d})^{(k)} (\frac{1}{1-x})^{(d-1-k)} \\
&= \sum_{k=0}^{d-1} \binom{d-1}{k} \frac{(d-1)!}{(d-1-k)!} x^{d-1-k} \frac{(d-1-k)!}{(d-1)!} (\frac{1}{1-x})^{d-k} \\
&- \sum_{k=0}^{d-1} \binom{d-1}{k} \frac{(n+d)!}{(n+d-k)!} x^{n+d-k} \frac{(d-1-k)!}{(d-1)!} (\frac{1}{1-x})^{d-k} \\
&= \sum_{k=0}^{d-1} \binom{d-1}{k} (\frac{x}{1-x})^{d-k-1} \frac{1}{1-x} \\
&- x^{n+1} \sum_{k=0}^{d-1} \binom{n+d}{k} (\frac{x}{1-x})^{d-k-1} \frac{1}{1-x}
\end{aligned}$$

If we now evaluate this equation at $x = 2$, we get:

$$(-1)^d + 2^{n+1} \cdot \sum_{k=0}^{d-1} \binom{n+d}{k} (-2)^{d-k-1}$$

If we assume a constant d and a large enough n, the binomial coefficient for $k = d-1$ becomes largest, making above expression $< 2^{n+1} \cdot \binom{n+d}{d-1} \sum_{k=0}^{d-1}(-2)^{d-k-1}$, where the final sum is in $\mathcal{O}(2^d)$, cancelling the 2^{-d} before the sum.

For the complexity this yields

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

showing that the number of grid points is in $\mathcal{O}(2^n \cdot n^{d-1})$ or, again using $h = 2^{-n}$, in $\mathcal{O}(h^{-1}log(h^{-1})^{d-1})$.

One can show that the interpolation error is in $\mathcal{O}(h^2 log(h^{-1})^{d-1})$ using sparse grids while for a single grid the error is in $\mathcal{O}(h^2)$ [Gar13].

2.4 Combination Technique

While sparse grids reduce the number of grid points significantly, they do require more complicated algorithms to work with. However, as we will see, a sparse grid function can be constructed by linearly combining a number of functions on single grids. The technique was first published for two dimensional grids in [GSZ92].

Let $f_{\vec{l}} \in V_{\vec{l}}$ denote a function on a single grid with level $\vec{l}$. The combination technique takes the grid functions from all grids with $n - d + 1 \leq |\vec{l}|_1 \leq n$ and combines them according to

$$f_n^c := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{|\vec{l}|_1 = n-q} f_{\vec{l}}$$

In this formula, functions on $W_{\vec{k}}$ appear exactly once iff $|\vec{k}|_1 \leq n$ and otherwise they do not appear at all. I will show this exemplary for $d = 2$. The combination technique then becomes $f_n^c = \sum_{|\vec{l}|_1=n} f_{\vec{l}} - \sum_{|\vec{l}|_1=n-1} f_{\vec{l}}$. That functions on $W_{\vec{k}}$ with $|\vec{k}|_1 > n$ do not appear follows from the involved single grid functions and the hierarchical redefinition of $V_{\vec{l}}$. Functions on difference spaces with $i = |\vec{k}|_1 \leq n$ appear exactly

$$\sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \binom{n-i-q+d-1}{d-1}$$

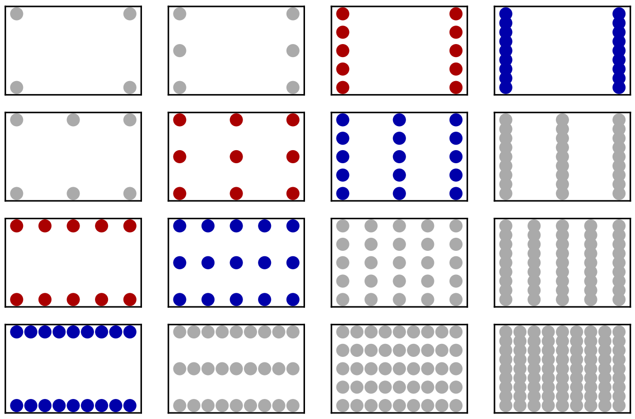


Figure 2.4: Considered grids for the combination technique with $d = 2$ and $n = 3$. Functions on blue grids are added, while functions on red grids get subtracted.

times. Specifically, for $d = 2$:

$$\begin{aligned}
&\sum_{q=0}^{2-1}(-1)^q\binom{2-1}{q}\binom{n-i-q+2-1}{2-1}\\
&= 1\cdot\binom{n-i-0+2-1}{1} - 1\cdot\binom{n-i-1+2-1}{1}\\
&= (n-i+1)-(n-1) = 1
\end{aligned}$$

This kind of proof extends into higher dimensions. Therefore the combination technique produces a solution for a sparse grid.

One can use this to numerically solve a PDE by solving the PDE on regular grids, then combining the solutions according to the above formula. In general, the solution obtained when using the combination technique is not equal to the sparse grid solution. However, under certain conditions, the error will again be the same as for sparse grids (For details see [GH14]).

2.5 CFL-Condition

When using an explicit solving method, certain criteria have to be met in order to guarantee convergence as the mesh width decreases. We will see that there exists an

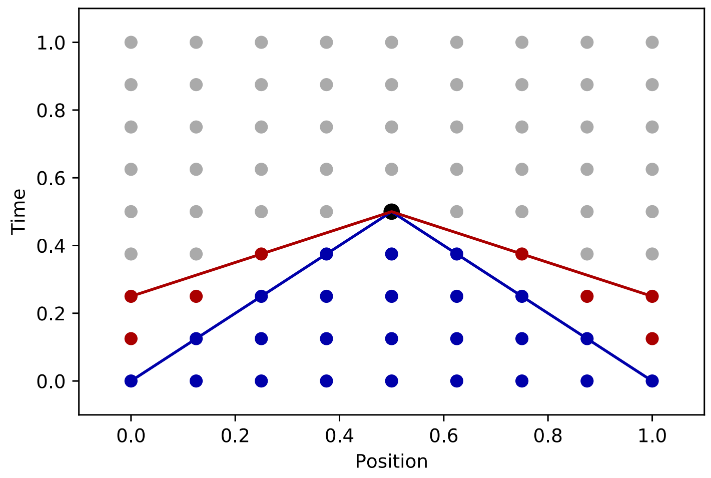


Figure 2.5: One dimensional grid with an additional axis for time. As the mesh width decreases the black grid point will become aware of every point in space below the blue line. Assuming information flows with a velocity depicted by the red line, a change in the function in the area between the red and blue lines will cause a change in the solution to the PDE, but not to the grid solution at the black point.

upper limit for timesteps h_t depending on the spatial mesh widths h_{x_i}. This section is based on the findings and formulations by Courant, Friedrichs, and Lewis [CFL28].

2.5.1 One spatial dimension

Assume that the PDE models some kind of information flow. Typical examples of such PDEs include advection and wave propagation. Assume that the information travels with any speed up to v in either direction. When the problem is discretized this means that a gridpoint at position x at time t has to be able to convey information to a point at position $x + \Delta x$ at time $t + \Delta t$ if $|\Delta t| \cdot v \geq |\Delta x|$, i.e. it can influence the new point. For convergence, this criteria has to be met for every pair of points as the mesh width approaches zero. A grid which does not fulfill this criteria is depicted in figure 2.5.

The finite difference method incorporates gridpoints up to l times the mesh width to the left and r times the mesh width to the right of the point to be integrated to the next timestep. This means that as the mesh width decreases, the maximum speed at which information flows throughout our grid is bounded by $l \cdot h_x / h_t$ for flows towards the right and by $r \cdot h_x / h_t$ towards the left. With a maximum information speed of v in either direction, we therefore have to meet $min(r,l) \geq \frac{v \cdot h_t}{h_x}$.

N.B. the left side of the equation is characteristic for the used method and is often abbreviated by the so called Courant-Number C, making the equation

$$C \geq \frac{v \cdot h_t}{h_x}$$

Notice that this means that grids with a smaller mesh width in the spatial dimensions require smaller timesteps.

2.5.2 Multiple dimensions

The idea of requiring the Volume of influence on one point in the PDE to be a subset of the volume of influential gridpoints as the mesh width approaches zero extends naturally into higher dimensions.

Assume that for every direction $\overrightarrow{\Delta x}$ information flows with a maximum speed of $v_{\overrightarrow{\Delta x}}$. To guarantee convergence even in the worst case, assume that some points are able to emit information in every direction simultaneously with maximum speed. Due to the latter assumption, the volume V_c of influence in the PDE is convex and scales with the time to the point in question.

Given a kernel of gridpoints in the current timestep to consider for a grid point in the next timestep, the Volume of influential grid points as the mesh width approaches zero (V_d) is simply a cone with a base similar to the convex hull of the kernel.

For convergence we now require

$$V_c \subseteq V_d$$

For the remainder of this thesis I will assume that information flows with a maximum speed of v_i in the i-th cardinal direction. Additionally I will assume the convex hull of influential grid points on point $\overrightarrow{p}$ to be evaluated for the next timestep to be $\{\overrightarrow{x} | \sum_{i=1}^{d} |\frac{(x_i - p_i)}{h_{x_i}}| \leq C\}$ (depicted for $C = 2$ in figure 2.6), where C is again given by the choice of influential grid points. This leads to the CFL-condition

$$C \geq \sum_{i=1}^{d} \frac{v_i \cdot h_t}{h_{x_i}}$$

Alternatively one may use an area of influence e.g. like $\{\overrightarrow{x} | \ \forall i \in [d]. |\frac{(x_i - p_i)}{h_{x_i}}| \leq C\}$ (depicted for $C = 1$ in figure 2.7), leading to the CFL-condition

$$C \geq \max_{i \in [d]} \frac{v_i \cdot h_t}{h_{x_i}}$$

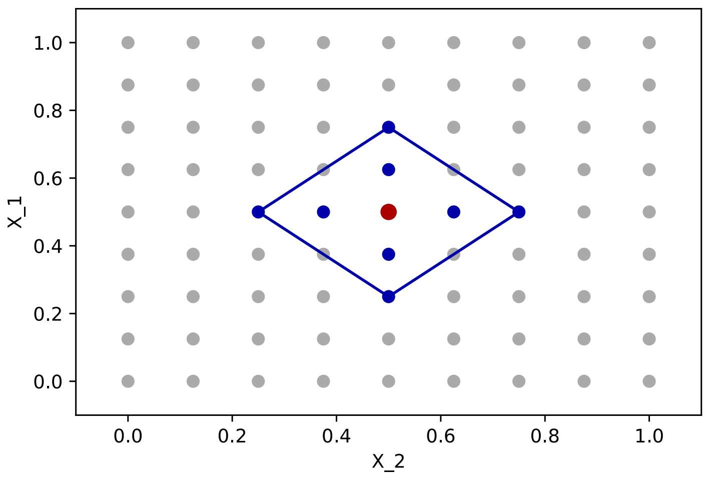


Figure 2.6: Diamond shaped area of influence for a kernel with $C = 2$ depicted by the blue dots. The red dot is to be integrated to the next timestep.

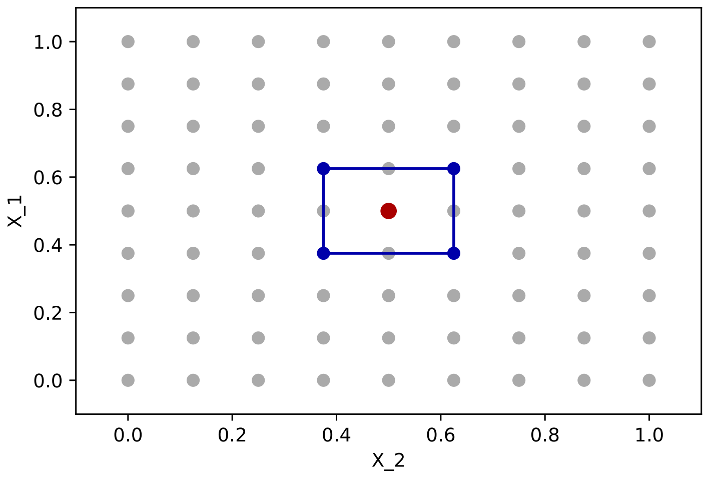


Figure 2.7: Square area of influence for a kernel with $C = 1$ depicted by the blue dots. The red dot is to be integrated to the next timestep.

Leading order results will not differ significantly, since for a fixed dimension d, the maximum timesteps which comply with these conditions will differ by not more than a constant factor (Which may however not necessarily be independent of d).

3 Time Stepping Strategies

We have seen, that under certain conditions we can efficiently construct a solution to a PDE by combining a number of regular grids to a sparse grid. However, we have also seen that in order to allow for convergence, we have to choose the timestep for integration below a certain threshold depending on the mesh widths of a grid. Since the grids needed for the combination technique have different widths, it might be possible to save computational resources by optimally choosing the timestep.

3.1 Uniform Timesteps

A common choice is to use a timestep, which allows for convergence on every grid involved. To evaluate the time needed to integrate the grid function, we consider the number of grid point updates. Every grid is updated once every timestep. This means that in leading order for the combination technique in order to advance a grid from time 0 to time 1,

$$\sum_{|\vec{l}|_1=n,n-1,\dots n-d+1} \frac{2^{|\vec{l}|_1}}{\Delta t_{\vec{l}}}$$

cell updates are needed. For uniform timesteps on all grids, i.e. $\Delta t_{\vec{l}} = \Delta t$, the number of cell updates is given by

$$\begin{aligned}
\mathbf{C}_u &= \frac{1}{\Delta t} \sum_{|\vec{l}|_1=n,n-1,\dots,n-d+1} 2^{|\vec{l}|_1} \\
&= \frac{1}{\Delta t} \sum_{q=0}^{d-1} \sum_{|\vec{l}|_1=n-q} 2^{n-q} \\
&= \frac{1}{\Delta t} \sum_{q=0}^{d-1} 2^{n-q} \sum_{|\vec{l}|_1=n-q} 1
\end{aligned}$$

The number of level-vectors having a manhattan norm of $n-q$ is given by $\binom{n-q+d-1}{d-1}$.

$$
\begin{aligned}
&= \frac{1}{\Delta t}\sum_{q=0}^{d-1} 2^{n-q}\binom{n-q+d-1}{d-1} \\
&= \frac{1}{\Delta t}\sum_{q=0}^{d-1} \mathcal{O}\Big(2^n \cdot \frac{(n-q+d-1)!}{(d-1)!(n-q)!}\Big) \\
&= \frac{1}{\Delta t}\sum_{q=0}^{d-1} \mathcal{O}\Big(2^n \cdot (\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2}))\Big) \\
&= \frac{1}{\Delta t}\sum_{q=0}^{d-1} \mathcal{O}\Big(2^n \cdot n^{d-1}\Big) \\
&= \frac{1}{\Delta t}\mathcal{O}\Big(d \cdot 2^n \cdot n^{d-1}\Big)
\end{aligned}
$$

If we again set $h := 2^{-n}$, we have

$$= \frac{1}{\Delta t}\mathcal{O}\Big(d \cdot h^{-1} log(h^{-1})^{d-1}\Big)$$

And finally the CFL-condition requires at least $\Delta t \in \Theta(h)$, showing

$$= \mathcal{O}\Big(d \cdot h^{-2} log(h^{-1})^{d-1}\Big)$$

3.2 Optimal Timesteps

To use the minimal amount of computational resources, and therefore to minimize the number of cell updates, we maximize the timestep on any grid with level $\vec{l}$ according to the CFL-condition

$$C \geq \sum_{i=1}^{d} \frac{v_i \cdot \Delta t_{\vec{l}}}{2^{-l_i}}$$

which I make tight and rearrange to

$$\Delta t_{\vec{l}} = \frac{C}{\sum_{i=1}^{d} v_i \cdot 2^{l_i}}$$

The number of cell updates is then given by

$$\begin{aligned}
\mathbf{C}_o &= \sum_{|\vec{l}|_1 = n, n-1, \ldots, n-d+1} \frac{2^{|\vec{l}|_1}}{\Delta t_{\vec{l}}} \\
&= \sum_{q=0}^{d-1} \sum_{|\vec{l}|_1 = n-q} \frac{2^{n-q}}{\Delta t_{\vec{l}}} \\
&= \frac{1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{|\vec{l}|_1 = n-q} \sum_{i=1}^{d} v_i 2^{l_i} \\
&= \frac{1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{i=1}^{d} v_i \sum_{|\vec{l}|_1 = n-q} 2^{l_i} \\
&= \frac{1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{i=1}^{d} v_i \sum_{b=0}^{n-q} 2^{b} \sum_{|\vec{l}|_1 = n-q \wedge v_i = b} 1
\end{aligned}$$

The number of level-vectors having a fixed coordinate is equal to the number of $d-1$-dimensional vectors with manhattan norm $n-q-b$:

$$\begin{aligned}
&= \frac{1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{i=1}^{d} v_i \sum_{b=0}^{n-q} 2^{b} \binom{n+d-q-b-2}{n-q-b} \\
&= \frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{b=0}^{n-q} 2^{b} \binom{n+d-q-b-2}{n-q-b}
\end{aligned}$$

Substituting $x := n-q-b$ for easier reading:

$$\begin{aligned}
&= \frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} \sum_{x=0}^{n-q} 2^{2n-2q-x} \binom{x+d-2}{x} \\
&= \frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} 2^{2n-2q} \sum_{x=0}^{n-q} \left(\frac{1}{2}\right)^{x} \binom{x+d-2}{d-2}
\end{aligned}$$

Repeating the technique already used by [Gar13] for shoing the number of grid points in a sparse grid, I will again represent the latter sum as a function $f(y)$ evaluated this time at $y = 1/2$. I.e.

$$\begin{aligned}
f(y) &= \sum_{x=0}^{n-q} y^x \binom{x+d-2}{d-2} \\
&= \frac{1}{(d-2)!} \sum_{x=0}^{n-q} (y^{x+d-2})^{(d-2)} \\
&= \frac{1}{(d-2)!} \Big(y^{d-2} \cdot \frac{1-y^{n-q+1}}{1-y} \Big)^{(d-2)} \\
&= \frac{1}{(d-2)!} \sum_{k=0}^{d-2} \binom{d-2}{k} (y^{d-2} - y^{d+n-q-1})^{(k)} \Big(\frac{1}{1-y} \Big)^{(d-2-k)} \\
&= \sum_{k=0}^{d-2} \binom{d-2}{k} \frac{(d-2)!}{(d-2-k)!} y^{d-2-k} \frac{(d-2-k)!}{(d-2)!} \Big(\frac{1}{1-y} \Big)^{d-2-k+1} \\
&- \sum_{k=0}^{d-2} \binom{d-2}{k} \frac{(d-1+n-q)!}{(d-1+n-q-k)!} y^{d-1+n-q-k} \frac{(d-2-k)!}{(d-2)!} \Big(\frac{1}{1-y} \Big)^{d-2-k+1} \\
&= \sum_{k=0}^{d-2} \binom{d-2}{k} \Big(\frac{y}{1-y} \Big)^{d-2-k} \cdot \Big(\frac{1}{1-y} \Big) \\
&- y^{n-q+1} \sum_{k=0}^{d-2} \binom{d-1+n-q}{k} \Big(\frac{y}{1-y} \Big)^{d-2-k} \cdot \Big(\frac{1}{1-y} \Big)
\end{aligned}$$

Evaluating this expression at $y = 1/2$:

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

Substituting this expression back into the number of cell updates:

$$\begin{aligned}
&\frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} 2^{2n-2q} \Big(2^{d-1} - 2^{-n+q} \sum_{k=0}^{d-2} \binom{d-1+n-q}{k} \Big) \Big) \\
=&\frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} 2^{2n-2q+d-1} - \frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{k=0}^{d-2} \binom{d-1+n-q}{k}
\end{aligned}$$

The first term has a complexity of $\mathcal{O}(2^{2n+d})$ while the latter (negative) term is of complexity $\mathcal{O}(2^n n^{d-2})$ (proof omitted). Therefore the number of cell updates needed when choosing the maximum possible timesteps for each grid is in $\mathcal{O}(2^d h^{-2})$

3.3 2-Power Timesteps

Since the mesh widths differ only in powers of 2, one might try to do the same for timesteps to prevent the need for any time synchronization methods for the combination. In two dimensions [LKV01] has already shown that using this strategy a complexity of $\mathcal{O}(h^{-2})$ can be achieved. More specifically, I will evaluate the cost of a time stepping strategy given by

$$\Delta t_{\vec{l}} = \frac{C}{|\vec{v}|_1 \cdot \max\limits_{i\in[d]} 2^{l_i}} \leq \frac{C}{\sum_{i=1}^{d} v_i \cdot 2^{l_i}}$$

I will again consider the number of cell updates:

$$\begin{aligned}
\mathbf{C}_p &= \sum_{|\vec{l}|_1 = n, n-1, \ldots, n-d+1} \frac{2^{|\vec{l}|_1}}{\Delta t_{\vec{l}}} \\
&= \sum_{q=0}^{d-1} \sum_{|\vec{l}|_1 = n-q} \frac{2^{n-q}}{\Delta t_{\vec{l}}} \\
&= \sum_{q=0}^{d-1} \sum_{|\vec{l}|_1 = n-q} 2^{n-q} \frac{|\vec{v}|_1}{C} \max_{i\in[d]} 2^{l_i} \\
&= \frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} \sum_{|\vec{l}|_1 = n-q} 2^{n-q} 2^{|\vec{l}|_\infty} \\
&= \frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{|\vec{l}|_1 = n-q} 2^{|\vec{l}|_\infty} \\
&= \frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{i=\lceil \frac{n-q}{d} \rceil}^{n-q} 2^i \sum_{|\vec{l}|_1 = n-q \wedge |\vec{l}|_\infty = i} 1
\end{aligned}$$

Vectors with a maximum norm of i make up the surface of a d-dimensional axis-aligned hypercube with line segments of length $2i$ centered around the origin. Vectors with a manhattan norm of $n-q$ lie in the $d-1$ dimensional hyperplane through the

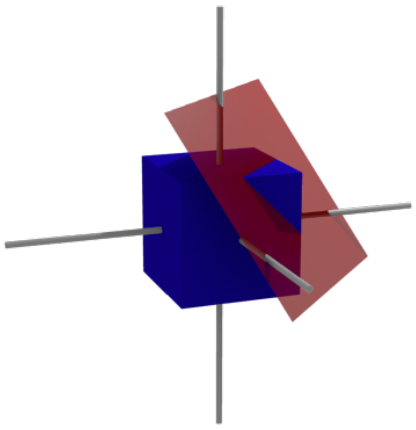


Figure 3.1: Vectors with an infinity norm of 1 (blue, only the surface of the cube) and with a manhattan norm of 2 (red). We are interested in the intersection of both.

points $(n-q)\cdot\vec{e_t},\ t\in[d]$, where $\vec{e_t}$ is the t-th unit vector. Naturally vectors which satisfy both conditions must lie in the hypervolume of the intersection of both. An example in 3 dimensions is depicted in figure 3.1.

Since we are only interested in intersection-vectors with non-negative entries (which is why i goes from $\left\lceil\frac{n-q}{d}\right\rceil$ to $n-q$), the hyperplane slices the cube into two parts, one of which is a d-dimensional simplex defined by its vertex set V with elements

$$\begin{aligned} v_0 &:= \vec{i} \\ v_t &:= \vec{i} - (di-n+q)\vec{e_t} && t\in[d] \end{aligned}$$

which has right angles at v_0. For ease of presentation I am going to mirror this simplex at $v_0/2$, to redefine the vertex set to

$$\begin{aligned} v_0 &:= \vec{0} \\ v_t &:= (di-n+q)\vec{e_t} && t\in[d] \end{aligned}$$

The hypervolume of the intersection is equal to the combined hypervolumes of the simplices with vertecies $\binom{V\setminus v_0}{d-2}$. There are d such simplices with equal hypervolume. To measure the volume of each, we take a look at the $d-1$-dimensional simplex defined by the vertex set $W := V \setminus v_d$. Let A_X be the hypervolume of a simplex with vertex set X. Since we have only right angles at v_0, it holds:

$$A^2_{W\setminus v_0} = \sum_{t=1}^{d-1} A^2_{W\setminus v_t}$$

Again, the hypervolumes on the righthand side are equal:

$$A^2_{W\setminus v_0} = (d-1)A^2_{W\setminus v_{(d-1)}}$$

We can calculate the volume of the simplex with vertex set $W \setminus v_{d-1}$ as

$$\begin{aligned} A_{W\setminus v_{d-1}} &:= \frac{1}{(d-2)!} \cdot det \begin{bmatrix} di-n+q & 0 & \dots & 0 \\ 0 & di-n+q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & di-n+q \end{bmatrix} \\ &= \frac{(di-n+q)^{d-2}}{(d-2)!} \end{aligned}$$

Putting all of this together, the volume of the intersection becomes

$$d \cdot \sqrt{d-1} \cdot \frac{(di-n+q)^{d-2}}{(d-2)!}$$

If we assume that the number of level vectors within this volume has the same complexity as the volume itself, we can substitute this back into our formula:

$$\frac{|\vec{v}|_1}{C} \sum_{q=0}^{d-1} 2^{n-q} \sum_{i=\lceil \frac{n-q}{d} \rceil}^{n-q} 2^i d \cdot \sqrt{d-1} \cdot \frac{(di-n+q)^{d-2}}{(d-2)!}$$

Since we are mainly interested in the complexity in n, it suffices to consider the largest summand, which is the one for $q = 0$. (Worst case we have an additional factor of d.) The term then becomes

$$\frac{|\vec{v}|_1}{C} 2^n \sum_{i=\lceil \frac{n}{d} \rceil}^{n} 2^i d \cdot \sqrt{d-1} \cdot \frac{(di-n)^{d-2}}{(d-2)!}$$

which is bounded from below by the summand for $i = n$:

$$\frac{|\vec{v}|_1}{C} 2^{2n} d \cdot \sqrt{d-1} \cdot \frac{(d-1)^{d-2} n^{d-2}}{(d-2)!}$$

and from above by:

$$\begin{aligned}
&\frac{|\vec{v}|_1}{C} 2^n \sum_{i=\lceil \frac{n}{d} \rceil}^{n} 2^i d \cdot \sqrt{d-1} \cdot \frac{(d-1)^{d-2} n^{d-2}}{(d-2)!} \\
=&\frac{|\vec{v}|_1}{C} 2^n d \cdot \sqrt{d-1} \cdot \frac{(d-1)^{d-2} n^{d-2}}{(d-2)!} \sum_{i=\lceil \frac{n}{d} \rceil}^{n} 2^i \\
\leq&\frac{2|\vec{v}|_1}{C} 2^{2n} d \cdot \sqrt{d-1} \cdot \frac{(d-1)^{d-2} n^{d-2}}{(d-2)!}
\end{aligned}$$

showing that the complexity of the number of cell updates with a 2-power timestepping policy is in $\mathcal{O}(f(d) \cdot 2^{2n} \cdot n^{d-2})$, or in $\mathcal{O}(f(d) \cdot h^{-1} log(h^{-1})^{d-2})$.

3.4 Summary

As we have seen, exploiting the CFL-limitations for every grid seperately can decrease the complexity from $\mathcal{O}(h^{-2} log(h^{-1})^{d-1})$ to only $\mathcal{O}(h^{-2})$. This reduction to the complexity to two dimensions comes however at the price of a factor of 2^d instead of only d, which might become significant for low resolution grids in high dimensions.

In general, the maximum timesteps given by the CFL condition might have non-rational ratios. To allow for a combination with low error rates in the combination technique, it is however necessary, that all involved grids have been advanced to the same time t_e. There are a number of workarounds, each having their benefits:

- End integration with a possibly smaller timestep (easy to implement)
- End integration early and extrapolate the gridpoints at t_e (possible cost savings where the integration steps are very costly)
- Do an additional integration step beyond t_e and interpolate the grid at t_e using q-th order interpolation (small errors may get smoothed out)

- Calculate the minimum number of steps beforehand and devide the time-domain equally (possibly less error due to the lack of large timesteps when doing only a few timesteps)

None of these workarounds provide an increase in complexity, since the number of cell updates for one additional timestep on every involved grid is in $\mathcal{O}(h^{-1}log(h^{-1})^{d-1})$

In two-dimensional grids, there is also the option of choosing a constant times the highest possible power of two for every grid. This means, that for those grids no further time synchronization steps are necessary, since at regular time intervals all grids are at the same time and can be combined. In higher dimensional grids however, as we have seen, we do get additional logarithmic factors, since the complexity for this strategy is in $\mathcal{O}(h^{-1}log(h^{-1})^{d-2})$. Still this strategy might be useful for small enough n.

So far we have assumed known maximum velocities in every cardinal direction. If this assumption does not hold one must adjust the needed timesteps at runtime. Using an explicit method this might become a problem since, depending on the PDE, the maximum velocity within one timestep might be hard to calculate.

4 Numerical Results

This chapter is organized as follows:

First I will test the complexity for the number of cell updates for moderate n, then I will compare the error for some time synchronization strategies, and lastly I will test the benefits of dynamic timestep adjustments.

4.1 Complexity test

For this test I have run the combination technique for $d = 2, 3, 5$ and several moderate n over the temporal interval $[0, 1]$ using smallend synchronizations (see next chapter) and recorded the number of cell updates.

To visualze the different complexities, the timesteps were chosen as follows:

- Uniform timesteps:$\frac{5}{2^n}$
- Optimal timesteps:$\frac{1}{\sum_{i \in [d]} 2^{l_i}}$
- 2-Power timesteps:$\frac{1}{2^{|\vec{l}|_\infty}}$

The results are plotted in figure 4.1.

Fitting these functions to reveal the exponent of the log-term gives complexities according to the table 4.1.

As expected, the optimal strategy was able to solve the PDE in $\mathcal{O}(h^{-2})$ cell updates regardless of the dimension.

strategy	Uniform	Optimal	2-Power
2d	$\mathcal{O}(2^{2n} \cdot n)$	$\mathcal{O}(2^{2n})$	$\mathcal{O}(2^{2n})$
3d	$\mathcal{O}(2^{2n} \cdot n^{1.5})$	$\mathcal{O}(2^{2n})$	$\mathcal{O}(2^{2n})$
5d	$\mathcal{O}(2^{2n} \cdot n^{2.5})$	$\mathcal{O}(2^{2n})$	$\mathcal{O}(2^{2n} \cdot n^{0.5})$

Table 4.1: The complexity of functions fitted onto the number of cell updates in figure 4.1.

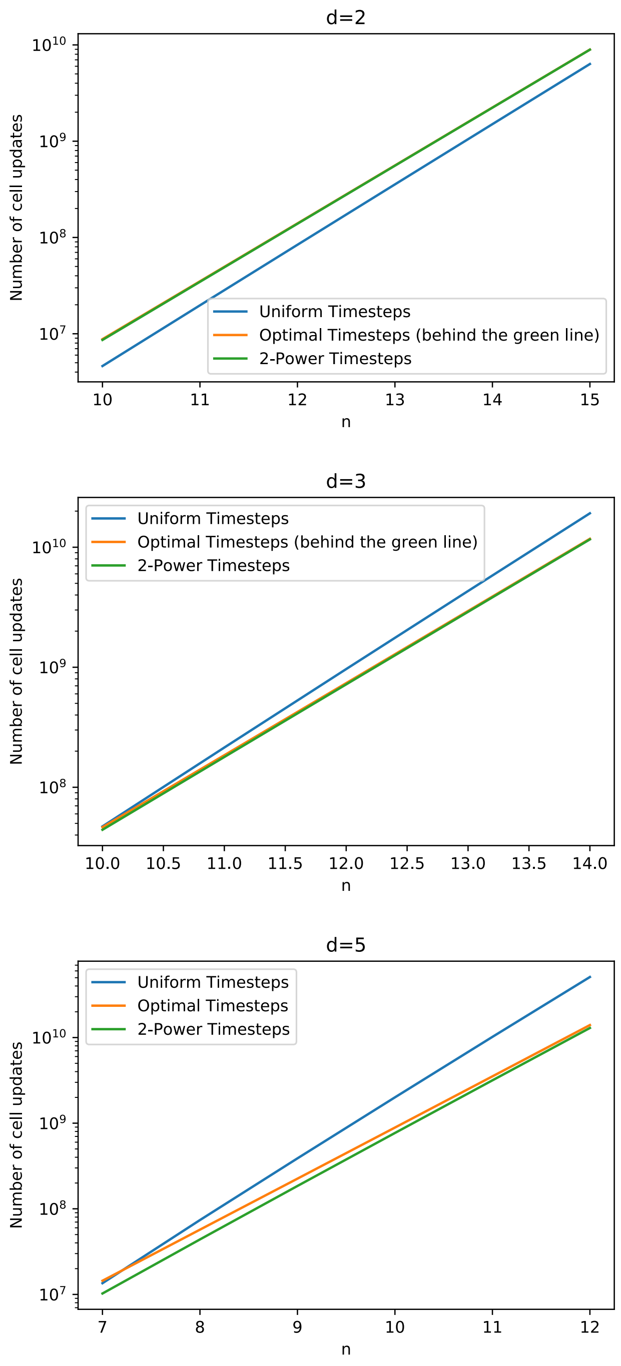


Figure 4.1: The number of cell updates measured. The timesteps were chosen to visualize the different complexities so the graphs should not be compared in terms of actual values.

The uniform timestepping strategy showed a complexity like $\mathcal{O}(h^{-2}log(h^{-1})^{d/2})$ instead of the predicted $\mathcal{O}(h^{-2}log(h^{-1})^{d-1})$. Closer inspection of the term

$$\frac{1}{\Delta t}\sum_{q=0}^{d-1}\mathcal{O}\Big(2^n\cdot\frac{(n-q+d-1)!}{(d-1)!(n-q)!}\Big)$$

from section 3.1 however shows, that the leading term in n is attenuated by a factor of $\frac{1}{(d-1)!}$. This leads to the assumption that this term will become important for larger n only. For small enough n the usage of uniform timesteps might therefore still be feasable.

I have predicted the 2-Power timestepping strategy to have a complexity of $\mathcal{O}(h^{-2}log(h^{-1})^{d-2})$. The numerical results however have a significantly lower complexity for the tested combinations of d and n. My best guess is that the assumption that the number of level-vectors is of the same complexity as the volume of a simplex, i.e.

$$\sum_{|\vec{l}\,|_1=n-q\wedge|\vec{l}\,|_\infty=i}1=\mathcal{O}\Big(d\cdot\sqrt{d-1}\cdot\frac{(di-n+q)^{d-2}}{(d-2)!}\Big)$$

(taken from section 3.3) holds only for large enough n.

The results in the 5-dimensional case seem to confirm that for problems in higher dimensions the optimal time stepping strategy is to be preferred above the other two, and the uniform strategy is infeasable compared to the 2-Power timestepping strategy. At the cost of the need for time synchronization methods, the optimal time stepping strategy can reduce the complexity of the integration in any dimension to that of a problem in one spatial dimension. The 2-Power strategy may be used without increased complexity in 2 spatial dimensions. Additionally for moderate n it is feasable in slightly higher dimensions.

4.2 Synchronization error

This section is devoted to comparing different synchronization methods in terms of their produced error. Concretely, we will take a look at the twodimensional wave equation given by

$$\frac{\partial^2 u}{\partial t^2}=c^2\cdot\Big(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\Big)$$

where x and y are the spatial dimensions, t is the time-dimension, $u(x, y, t)$ the quantity we are interested in, and c the velocity of the waves. Transformation into a system of equations of the first order gives

$$\begin{aligned}\frac{\partial u}{\partial t} &= u_t \\ \frac{\partial u_t}{\partial t} &= c^2 \cdot \Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big)\end{aligned}$$

where u_t is the temporal derivative of u. This transformation prevents the need for grids at multiple timesteps caused by the second derivative in time. It does however come at the cost of halved optimal timesteps, since information on the u-grid does now need two timesteps to progress by one spatial mesh width. Using forward differences in time and central differences in space yields the following discretized system:

$$\begin{aligned}\frac{u_{0,0}^{t+1} - u_{0,0}^{t}}{h_t} &= (u_t)_{0,0}^{t} \\ \frac{(u_t)_{0,0}^{t+1} - (u_t)_{0,0}^{t}}{h_t} &= c^2 \cdot \Big(\frac{u_{1,0}^{t} - 2 \cdot u_{0,0}^{t} + u_{-1,0}^{t}}{h_x^2} + \frac{u_{0,1}^{t} - 2 \cdot u_{0,0}^{t} + u_{0,-1}^{t}}{h_y^2}\Big)\end{aligned}$$

where $F_{a,b}^{t}$ is shorthand for $F(x + a \cdot h_x, y + b \cdot h_y, t \cdot h_t)$.

Resolving for u and u_t at the new timestep:

$$\begin{aligned}u_{0,0}^{t+1} &= u_{0,0}^{t} + h_t \cdot (u_t)_{0,0}^{t} \\ (u_t)_{0,0}^{t+1} &= (u_t)_{0,0}^{t} + h_t \cdot c^2 \cdot \Big(\frac{u_{1,0}^{t} - 2 \cdot u_{0,0}^{t} + u_{-1,0}^{t}}{h_x^2} + \frac{u_{0,1}^{t} - 2 \cdot u_{0,0}^{t} + u_{0,-1}^{t}}{h_y^2}\Big)\end{aligned}$$

I will use the following Dirichlet boundary conditions

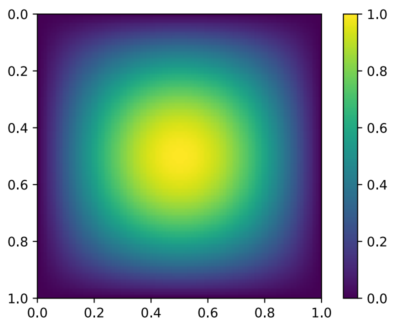
$$\begin{aligned}u(x, y, t) &= 0 \qquad &&, \{x, y\} \cap \{0, 1\} \neq \emptyset \\ u_t(x, y, t) &= 0 \qquad &&, \{x, y\} \cap \{0, 1\} \neq \emptyset\end{aligned}$$

and the initial functions

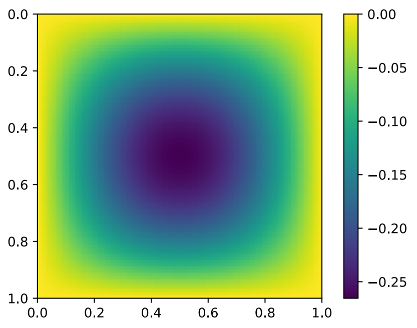
$$\begin{aligned}u(x, y, 0) &= sin(\pi \cdot x) \cdot sin(\pi \cdot y) \\ u_t(x, y, 0) &= 0.\end{aligned}$$

I am integrating from $t = 0$ to $t = 1$ with a velocity of $c = 1$. Figure 4.2 shows the initial grid functions as well as the solution obtained on a full grid of level $(7, 7)$ at a sufficiently small timestep.

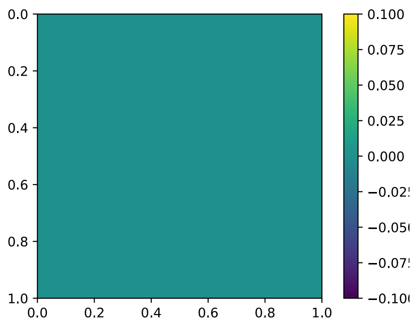
We are going to compare the error of the following four synchronization strategies:



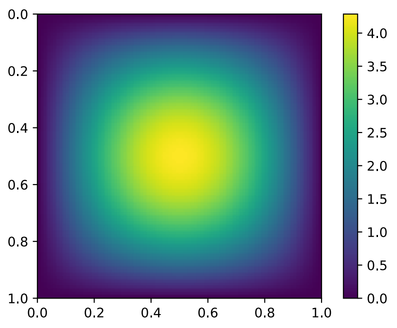
(a) u at $t = 0$



(b) u at $t = 1$



(c) u_t at $t = 0$



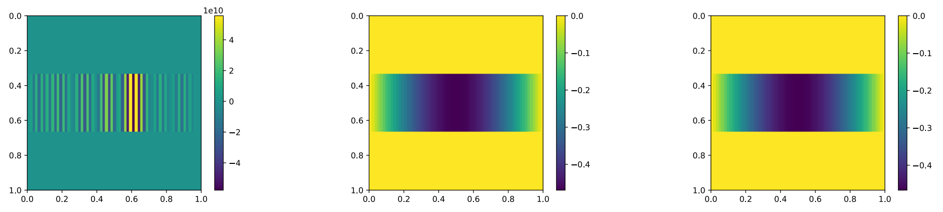
(d) u_t at $t = 1$

Figure 4.2: Both the u and u_t grids at times $t = 0, 1$.

- Smallend: Optimal timesteps are taken, where possible without integrating the grid beyond $t = 1$. Afterwards a smaller timestep is taken to reach $t = 1$
- Inter: Optimal timesteps are taken till the grid has passed $t = 1$. Afterwards the grid at $t = 1$ will be aquired by linearly interpolating the grids directly before and after $t = 1$.
- Extra: Optimal timesteps are taken as long as $t = 1$ would not be passed. Afterwards the grid at $t = 1$ will be aquired by linearly extrapolating it from the two previous grids.
- Uniform: The minimal number of necessary optimal timesteps k is calculated and the temporal domain devided into k timesteps, each of size $1/k$.

Figures 4.3a, 4.3c and 4.3e show the resulting differences from the full grid solution at level $(7, 7)$ for some small n with full optimal timesteps, $\frac{1}{10}$ and $\frac{1}{100}$ of optimal timesteps, aquired using the combination technique.

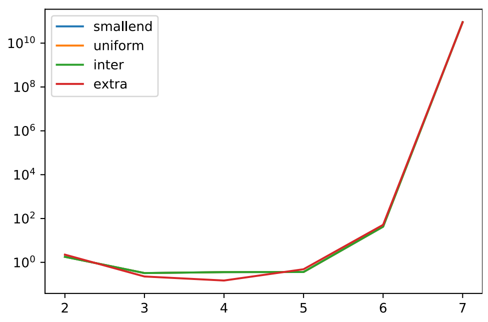
With increasing levels the solution seems to evolve some sort of instability, which is not caused by violation of the CFL-condition. Inspection of the subgrids for the combination technique shows that the error only occurs on the outermost grids. The following figure for example shows the full grid with level $(1, 6)$ aquired through integration using uniform timesteps based on 1 times, $\frac{1}{10}$ times and $\frac{1}{100}$ times the optimal timestep.



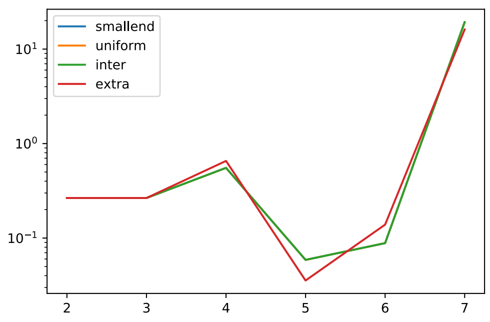
(a) optimal timesteps (b) $\frac{1}{10}$ of optimal timesteps (c) $\frac{1}{100}$ of optimal timesteps

Figure 4.4: Component grids of level $(1, 6)$ at $t = 1$ integrated using uniform timesteps.

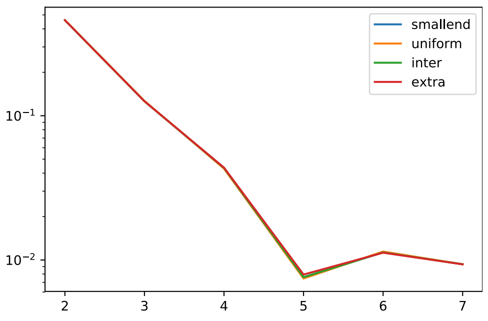
There is the possibility to leave out certain grids for the combination technique. The so-called truncated combination technique, which is defined more detailed in [Obe+17] leaves out all grids which have a level component below a given constant τ. Just as with the combination technique, the truncated combination technique counts every hierarchical subspace of a sparse grid exactly once.



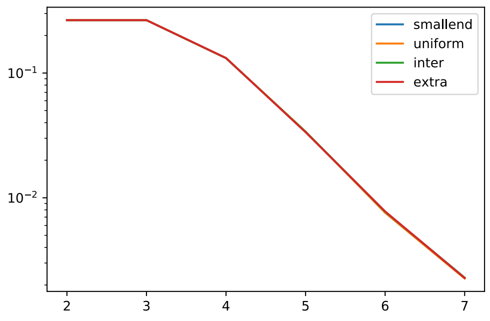
(a) optimal timesteps



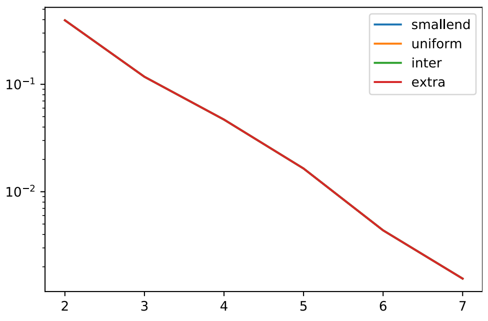
(b) optimal timesteps



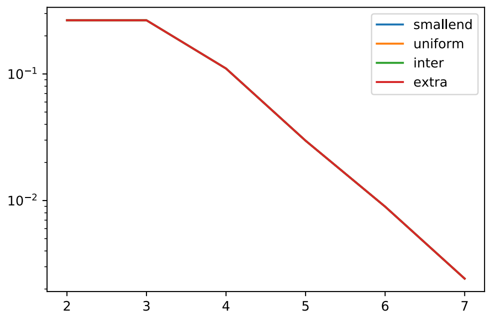
(c) $\frac{1}{10}$ of optimal timesteps



(d) $\frac{1}{10}$ of optimal timesteps



(e) $\frac{1}{100}$ of optimal timesteps



(f) $\frac{1}{100}$ of optimal timesteps

Figure 4.3: Errors on the u-grid obtained using the plain combination technique (on the left) and the truncated combination technique with $\tau = 2$ (on the right). The x-Axis denotes the level n.

N.B. [Obe+17] defines both the normal combination technique as well as the truncated combination technique slightly different. While both definitions use the same grids, this results in some index shifts in n.

Figures 4.3b, 4.3d and 4.3f measure the same error as figures 4.3a, 4.3c and 4.3e, but using the truncated combination technique with $\tau = 2$.

As either the number of grid points or the number of time steps increases, the difference between the synchronization methods becomes unnoticable small. Figures 4.3d and 4.3f demonstrate that spatial discretization error quickly takes over, deminishing every gain from altered timesteps. In figure 4.3c with $n = 5$ uniform timesteps showed a slight benefit, which can happen sometimes due to the lack of additional interpolation error and (at least with small n) too large timesteps.

The most prominent distinction can be made in figure 4.3b, where the extrapolated solution seems to differ notably from the other solutions. This is likely because of our redefinition of the second-order PDE into a system of first-order PDEs. Information flow over one mesh-width became a two-step process, allowing for two slightly different solutions at alternating positions in time. As can be expected, the effect disappears for larger n and/or smaller timesteps.

As a conclusion, the exact synchronization method does not influence the error significantly when choosing timesteps on high-precision grids. Smallend may be preferred because it does not require knowledge of the actual maximal velocities, and is in most cases easier to implement than methods involving inter- or extrapolation of grids.

4.3 Dynamic timestep adjustments

To evaluate the utility of choosing an optimal timestep at runtime we will take a look at the two-dimensional shallow water equations (SWE). They are given by

$$\begin{aligned}
\frac{\partial(\rho\eta)}{\partial t} + \frac{\partial(\rho\eta u)}{\partial x} + \frac{\partial(\rho\eta v)}{\partial y} &= 0 \\
\frac{\partial(\rho\eta u)}{\partial t} + \frac{\partial}{\partial x}\left(\rho\eta u^2 + \frac{1}{2}\rho g\eta^2\right) + \frac{\partial(\rho\eta uv)}{\partial y} &= 0 \\
\frac{\partial(\rho\eta v)}{\partial t} + \frac{\partial(\rho\eta uv)}{\partial x} + \frac{\partial}{\partial y}\left(\rho\eta v^2 + \frac{1}{2}\rho g\eta^2\right) &= 0
\end{aligned}$$

where x and y are the spatial dimensions, t is the time dimension, $\rho(x, y, t)$ the fluid density, $\eta(x, y, t)$ the fluid height, $\big(u(x, y, t), v(x, y, t)\big)$ a vector representing the fluid flow velocity along the x and y direction respectively, and $g(x,y,t)$ the gravitational constant at any given point.

For this thesis I will choose an uncompressible fluid and uniform time-independent gravitational acceleration (i.e. ρ and g are constant). Approximating derivatives in time with forward differences and derivatives in space with central differences, we get (writing $F^t_{o_1,o_2}$ instead of $F(x_1 + o_1 * h_x, x_2 + o_2 * h_y, t * h_t)$ for easier reading):

$$\frac{\eta^{t+1}_{0,0} - \eta^t_{0,0}}{h_t} + \frac{\eta^t_{1,0}u^t_{1,0} - \eta^t_{-1,0}u^t_{-1,0}}{2h_x} + \frac{\eta^t_{0,1}v^t_{0,1} - \eta^t_{0,-1}v^t_{0,-1}}{2h_y} = 0$$

$$\frac{\eta^{t+1}_{0,0}u^{t+1}_{0,0} - \eta^t_{0,0}u^t_{0,0}}{h_t} + \frac{\eta^t_{1,0}(u^t_{1,0})^2 + \frac{1}{2}g(\eta^t_{1,0})^2 - \eta^t_{-1,0}(u^t_{-1,0})^2 - \frac{1}{2}g(\eta^t_{-1,0})^2}{2h_x} + \frac{\eta^t_{0,1}u^t_{0,1}v^t_{0,1} - \eta^t_{0,-1}u^t_{0,-1}v^t_{0,-1}}{2h_y} = 0$$

$$\frac{\eta^{t+1}_{0,0}v^{t+1}_{0,0} - \eta^t_{0,0}v^t_{0,0}}{h_t} + \frac{\eta^t_{0,1}(v^t_{0,1})^2 + \frac{1}{2}g(\eta^t_{0,1})^2 - \eta^t_{0,-1}(v^t_{0,-1})^2 - \frac{1}{2}g(\eta^t_{0,-1})^2}{2h_y} + \frac{\eta^t_{1,0}u^t_{1,0}v^t_{1,0} - \eta^t_{-1,0}u^t_{-1,0}v^t_{-1,0}}{2h_x} = 0$$

where I have already left out the constant, non-influential fluid density. We can use the first formula to calculate the new fluid height, and the other formulas to calculate the new velocities like:

$$\eta^{t+1}_{0,0} = \eta^t_{0,0} - h_t \cdot \left(\frac{\eta^t_{1,0}u^t_{1,0} - \eta^t_{-1,0}u^t_{-1,0}}{2h_x} + \frac{\eta^t_{0,1}v^t_{0,1} - \eta^t_{0,-1}v^t_{0,-1}}{2h_y} \right)$$

$$u^{t+1}_{0,0} = \frac{\eta^t_{0,0}u^t_{0,0} - h_t \cdot \left(\frac{\eta^t_{1,0}(u^t_{1,0})^2 + \frac{1}{2}g(\eta^t_{1,0})^2 - \eta^t_{-1,0}(u^t_{-1,0})^2 - \frac{1}{2}g(\eta^t_{-1,0})^2}{2h_x} + \frac{\eta^t_{0,1}u^t_{0,1}v^t_{0,1} - \eta^t_{0,-1}u^t_{0,-1}v^t_{0,-1}}{2h_y} \right)}{\eta^{t+1}_{0,0}}$$

$$v^{t+1}_{0,0} = \frac{\eta^t_{0,0}v^t_{0,0} - h_t \cdot \left(\frac{\eta^t_{0,1}(v^t_{0,1})^2 + \frac{1}{2}g(\eta^t_{0,1})^2 - \eta^t_{0,-1}(v^t_{0,-1})^2 - \frac{1}{2}g(\eta^t_{0,-1})^2}{2h_y} + \frac{\eta^t_{1,0}u^t_{1,0}v^t_{1,0} - \eta^t_{-1,0}u^t_{-1,0}v^t_{-1,0}}{2h_x} \right)}{\eta^{t+1}_{0,0}}$$

To avoid error due to boundary conditions I will loop the grids (i.e. $F(x,y,t) = F(x+m, y+n, t)$ for $m, n \in \mathbb{N}, F \in \{\eta, u, v\}$). The initial functions will be

$$u(x,y,0) = v(x,y,0) = 0$$

$$n(x,y,0) = \max\{1, 1 + \frac{h}{r^2} \cdot (r^2 - 2rd + d^2)\} \quad , where\ d = \sqrt{(x-p_1)^2 + (y-p_2)^2}$$

with $h = 0.2$,$r = 0.2$,$p = (0.5, 0.5)^T$. The gravitational constant g is fixed with $g = 2$. We are going to integrate from $t = 0$ to $t = 0.3$. Figure 4.5 shows the initial grids and the solution on a single grid with level $(8, 8)^T$.

The maximum velocity over this time period in either direction is plotted below.

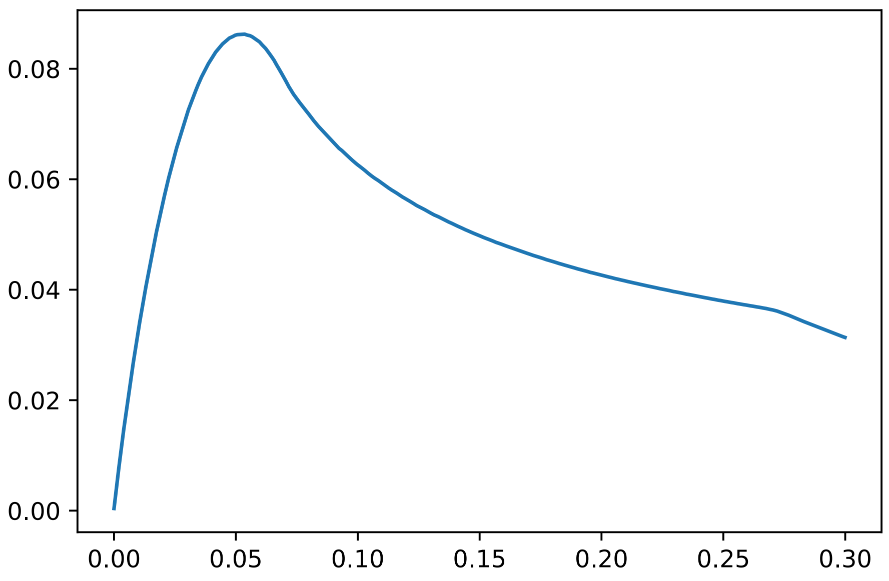


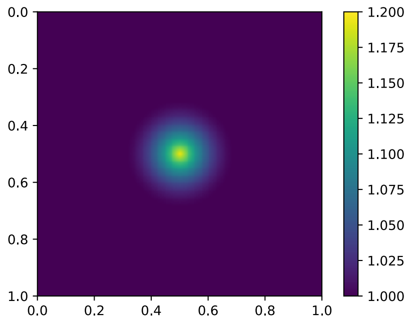
Figure 4.6: The maximal absolute velocity seen on u and v over time.

As one can see, the spike in the fluid height first accelerates its water mass due to gravity, then the waves, which travel outwards, eventually begin to slow down and finally at the end there is a slight bend downwards, when the mass with the highest velocity in any cardinal direction reaches the other side of the wave, as the grid is looping, thereby canceling its momentum.

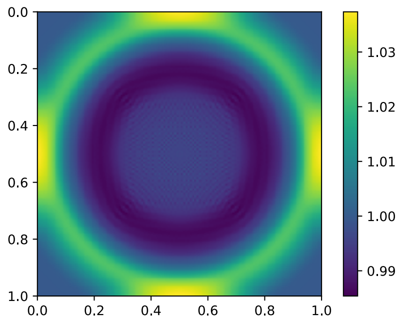
N.B. although the velocity stays below 0.09 and we are integrating over a time period of 0.3, the fluid mass seems to move a distance of about $0.4 - 0.5$. Although I was not able to find a reason for this, it may likely be a software bug. The following experiments should not be influenced in any other way than that a constant multiple of the *extra*-values will be required to reach the same level of accuracy.

We will compare the following two timestep policies:

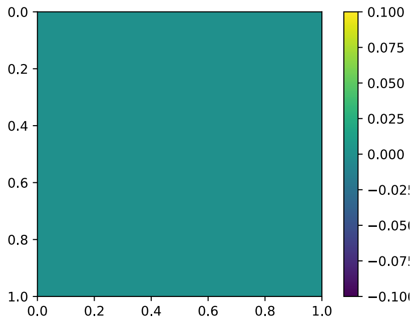
- static: We know that the velocity will stay below 0.09, so we can choose timesteps



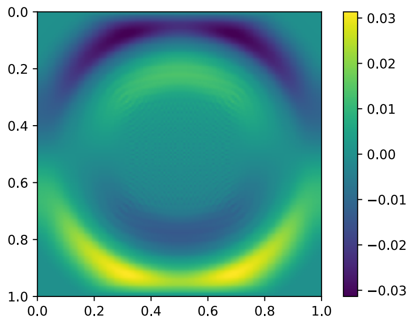
(a) η at $t = 0$



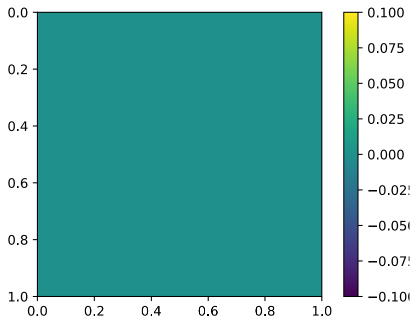
(b) η at $t = 0.3$



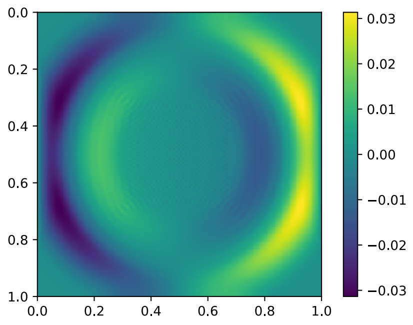
(c) u at $t = 0$



(d) u at $t = 0.3$



(e) v at $t = 0$



(f) v at $t = 0.3$

Figure 4.5: The initial grids and the grids at $t = 0.3$

as

$$\Delta t = \frac{1}{extra} \cdot \frac{1}{0.09/h_x + 0.09/h_y}$$

- dynamic: We can also decide to use any knowledge of the current velocities as

  $$\Delta t = \frac{1}{extra} \cdot \frac{1}{g_x/h_x + g_y/h_y}$$

  where

  $$g_x = \begin{cases} vel_{min}, max(u) < vel_{min} \\ vel_{max}, max(u) > vel_{max} \\ max(u), else \end{cases}$$

  $$g_y = \begin{cases} vel_{min}, max(v) < vel_{min} \\ vel_{max}, max(v) > vel_{max} \\ max(v), else \end{cases}$$

  and $max(u)$ and $max(v)$ represent the maximum values at the current time in the grids u and v respectively. Values get bounded from below by vel_{min} to prevent errors caused by the maximum velocity multiplying within one timestep. The upper bound, vel_{max}, is necessary to prevent grids developing infinitely high velocities due to instabilities, thereby preventing calculations from terminating. The following results used $vel_{max} = 2$ and $vel_{min} = 0.02$.

Since values close to the CFL-limitation tend to be unstable, I have introduced an additional factor $1/extra$ to adjust the precision of the grids. The following figure shows the time needed for calculations using above policies in dependency of $extra$:

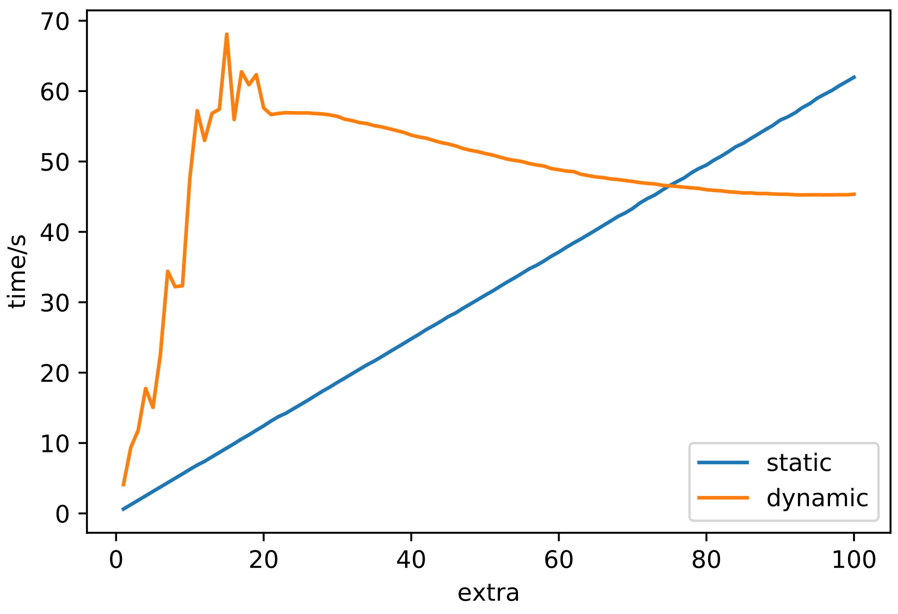
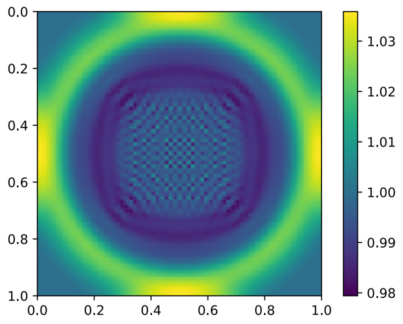


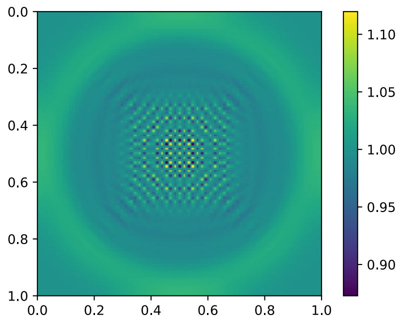
Figure 4.7: The time needed to integrate the grids from $t = 0$ to $t = 0.3$.

All calculations were made on a grid with level $(7,7)$ As can be expected, the static timestep policy took an amount of time proportional to the value of *extra*. More specifically, we can approximate the values by $0.62 \cdot extra$. With low values of *extra*, the grids using a dynamic timestep policy quickly developed large velocities, therefore taking a comparatively long time to be calculated. As *extra* increased, the grids became more stable, thereby adjusting to the velocities of the optimal solution, increasing the timesteps and eventually exceeding the average timestep length of the static policy.

However one is usually not interested in the computing time alone, but in the quality of the resulting grids. And as figure 4.8 shows, the dynamic timestep method creates a larger error than the static one for a given value of *extra*.



(a) using static timesteps



(b) using dynamic timesteps

Figure 4.8: The final grids with $extra = 100$

To compare the efficiency of the policies, for a given *extra*-value e_d using the dynamic policy, the time for the calculations is recorded. From that an estimate is made for an *extra*-value e_s for use with the static policy, which should take roughly the same amount of time. Afterwards the resulting grids are compared against a reference grid aquired on level $(7, 7)$ using a constant timestep of $1e - 6$ and the error is denoted in the following table:

e_d	computing time	approx e_s	Err_d	Err_s
200	7.115e1	114	0.013132	0.009160
500	1.752e2	282	0.002108	0.002081
1000	3.510e2	566	0.001970	0.001955
10000	3.465e3	5588	0.001831	0.001846

Table 4.2: Results for the efficiency comparison of timestep policies.

Only in the last case did the dynamic policy perform better than the static one. However, since these grids have about the same error as the grids with $e_d = 1000$, it is likely that this is due to noise. In general, the static timestep policy seems to be more suited for solving the PDE in a short amount of time.

This is not to say that dynamic policies can not be more efficient than static ones. A good policy will however depend on more than the CFL-limitation to prevent different types of error as they may occur depending on the precise problem and the used methods.

5 Conclusion

We have seen that by using the CFL-condition to optimize timesteps on every individual grid in the combination technique, one can reduce the complexity of any integration of PDEs with bounded information velocities from $\mathcal{O}(h^{-2}log(h^{-1})^{d-1})$ to just $\mathcal{O}(h^{-2})$, effectively making the complexity in the level n independent of the spatial dimension d.

The arising need for synchronization methods when using the combination technique can be satisfied by an easy to implement smaller timestep near the end of the integration without a significant loss in accuracy. With small values of n or in two dimensions it is even feasable to prevent the need for any synchronization methods by choosing timesteps with 2-power ratios.

When the maximum velocity is unknown or significantly higher than the average velocity (e.g. the speed of light in several physical phenomena), the timestep may be chosen dynamically based on the current maximum velocity. Using the presented method this may however impose additional computational work to achieve a given level of accuracy. Nevertheless, with a firm knowledge of the errors which may arise during integration, the method may possibly be adjusted to gain oppose this additional work and possibly even save computational resources.

One thing worth stressing is that the CFL-condition is not a sufficiency-condition for convergence. Depending on the problem there might be other conditions which have to be met in order to retrieve accurate results using an explicit method. E.g. the 1-dimensional heat equation is known to be unstable whenever $\frac{\alpha h_t}{h_x^2} > \frac{1}{2}$, which is a stronger limitation than the Courant-condition. Even without these effects one might want to consider using only a fraction of the timesteps permitted by the CFL-number to increase accuracy on grids with low levels.

List of Figures

List of Tables

Bibliography

[CFL28] R. Courant, K. Friedrichs, and H. Lewy. “Über die partiellen Differenzengleichungen der mathematischen Physik.” In: *Mathematische Annalen* 100 (1928), pp. 32–74. DOI: 10.1007/BF01448839.

[Gar13] J. Garcke. “Sparse grids in a nutshell.” In: *Sparse grids and applications*. Springer, 2013, pp. 57–80.

[GH14] M. Griebel and H. Harbrecht. “On the Convergence of the Combination Technique.” In: *Sparse Grids and Applications - Munich 2012*. Ed. by J. Garcke and D. Pflüger. Cham: Springer International Publishing, 2014, pp. 55–74. ISBN: 978-3-319-04537-5.

[GSZ92] M. Griebel, M. Schneider, and C. Zenger. *A Combination Technique For The Solution Of Sparse Grid Problems*. 1992.

[LKV01] B. Lastdrager, B. Koren, and J. Verwer. “The Sparse-Grid Combination Technique Applied To Time-Dependent Advection Problems.” In: *Appl. Numer. Math* 38 (2001), p. 2001.

[Obe+17] M. Obersteiner, A. P. Hinojosa, M. Heene, H.-J. Bungartz, and D. Pflüger. “A Highly Scalable, Algorithm-based Fault-tolerant Solver for Gyrokinetic Plasma Simulations.” In: *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*. ScalA ’17. Denver, Colorado: ACM, 2017, 2:1–2:8. ISBN: 978-1-4503-5125-6. DOI: 10.1145/3148226.3148229.

[Smo63] S. Smolyak. “Quadrature and interpolation formulas for tensor products of certain classes of functions.” In: *Soviet Mathematics, Doklady* 4 (1963), pp. 240–243.