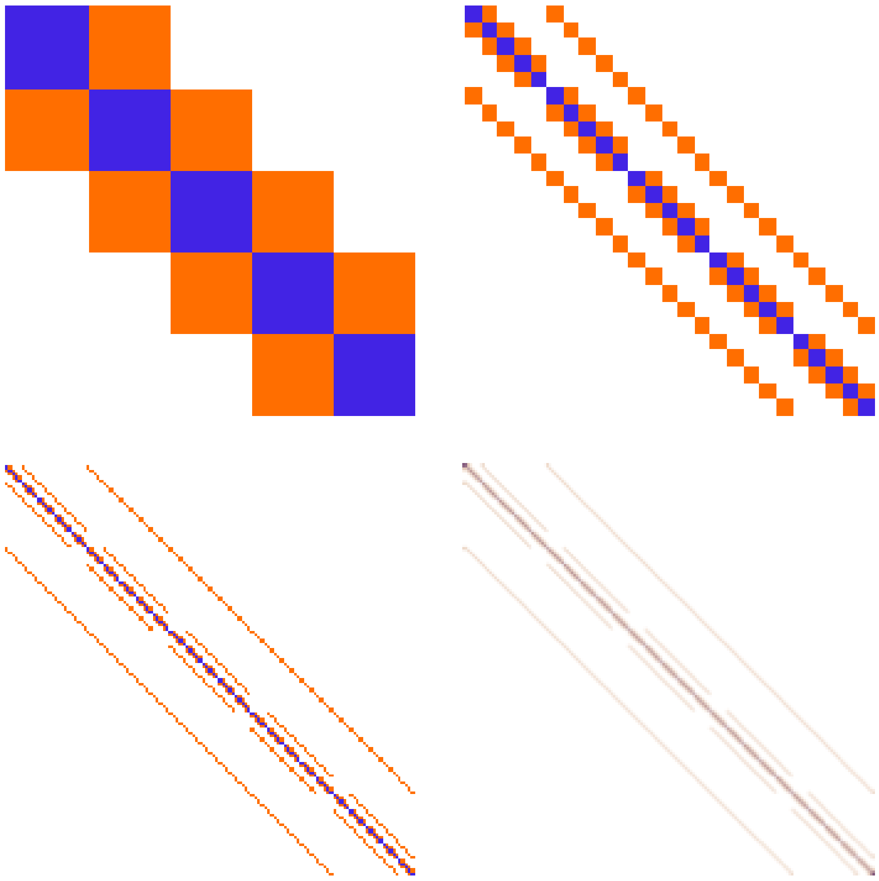
Adaptive d-dimensional Finite Difference Methods

Rouke Pouw



June 30th 2014

Supervisor:	Paul Zegeling
Second reader:	Rob Bisseling

Universiteit Utrecht

Contents

1. Introduction

In this thesis I will endeavor to solve higher dimensional initial boundary value problems using a generalized finite difference method. Boundary value problems arise in a very wide variety of research areas and although classical physical systems are usually limited to three space dimensions and a fourth time dimension applications in, for example, finances[**1**], molecular biology [**2**] and quantum physics [**3**] do include higher dimensional problems.

Increasing the dimension comes at a hefty cost. Challenges for higher dimensions are at first glance the exponential increase in memory and computation. But even before those the first hurdle is finding a fitting notation for the structures arising in these computations. The *curse of dimensionality* is in the notation as well.

I chose not to limit my approach to conventional uniform grids. Using the philosophy to first take a step back to then take a few steps forward I decided to define an approach for very generalized grids. This enables me to thoroughly analyze the arising structures and succeed in defining and applying uniform and r-refinement adaptive methods.

Where uniformity takes its strength from a highly structured homogenous approach adaptivity seeks to strategically improve problem areas. r-refinement is, opposed to other adaptive methods, a way to retain both the uniform advantages as strategic local benefits.

To facilitate the implementation of adaptive method I refrained from simplifying the method early on. Abandoning options beforehand may provide limitations later on. Confining to linear operators or setting boundaries to zero might simplify things for the uniform case but they are essential for r-refinement and might enable the application of even more creative structures that will very likely be needed in future improvements to the solving of higher dimensional boundary value problems.

CHAPTER 1

Higher dimensional Initial Boundary Value Problems

1. Method overview

1.1. Single system Method. An Initial-Boundary Value Problem, from here on shortened to Boundary Value Problem (BVP) as the initial conditions (ICs) can be considered as boundary conditions (BCs) in the time sense, consists of multiple Partial Differential Equations (PDEs). A very rough description of the method developed in this thesis is the following:

$$\text{(BVP)} \rightsquigarrow \left\{ \begin{array}{c} \text{(PDE)}_0 \\ \vdots \\ \text{(PDE)}_p \end{array} \right\} \rightsquigarrow \left\{ \begin{array}{c} \text{(FDE)}_0 \\ \vdots \\ \text{(FDE)}_p \end{array} \right\} \rightsquigarrow \left\{ \begin{array}{c} \text{(LS)}_0 \\ \vdots \\ \text{(LS)}_p \end{array} \right\} \rightsquigarrow \text{(LS)'} \tag{1.1}$$

(1.2) Graphical description of the complete method.

Each PDE is approximated using the finite difference method resulting in a Finite Difference Equation (FDE). These FDEs are then represented by Linear Systems (LS) which are combined into a larger Linear System (LS)'.

The process of approximating a PDE by a FDE consists of three parts: a finite difference discretization of the differential operator ($L \to S$), discretization of the domain ($\Omega \to W$), and discretization of the inhomogeneous term ($f \to \mathbf{f}$):

$$\begin{array}{ccccc} \text{(PDE)} & & \text{(FDE)} & & \text{(LS)} \\ & & & & \\ (L, \Omega, f) & \rightsquigarrow & (S, W, \mathbf{f}) & \rightsquigarrow & (A, \mathbf{b}) \end{array}$$

Graphical description of the method.

The matrix A and constant term $\mathbf{b}$ of the linear system are constructed from $(S, W, \mathbf{f})$. The solution of this linear system, $\mathbf{u}$, approximates the solution of (PDE), u, restricted to the nodes of the discretization.

1.2. Traversal. Especially for higher dimensional cases the linear system might end up being very large. Separating the system into multiple smaller systems provides options for more efficient solving. In this thesis I introduce the concept of *Traversal* to handle the dependency of these systems on each other. This approach is based on traversing through the unknowns layer by layer. Where time traversal is a very natural approach also non time dependent processes can benefit from a traversal that divides a larger linear system into multiple smaller ones.

The larger system is decoupled into multiple systems that can be ordered by their dependence. In other words: system $(\mathrm{LS})'_n$ only depends on values already solved from $(\mathrm{LS})'_0, \ldots, (\mathrm{LS})'_{n-1}$:

$$(\mathrm{BVP}) \rightsquigarrow \ldots \rightsquigarrow \left\{ \begin{matrix} (\mathrm{LS})_0 \\ \vdots \\ (\mathrm{LS})_p \end{matrix} \right\} \rightsquigarrow (\mathrm{LS})\text{'} \rightsquigarrow \left\{ \begin{matrix} (\mathrm{LS})\text{'}_0 \\ \vdots \\ (\mathrm{LS})\text{'}_N \end{matrix} \right\}$$

Graphical description of the traversal method.

Not only does the traversal cover the ordered decoupling of a linear system following from a PDE but also the sequence of solving the sub-PDEs of a BVP, starting with the lower dimensional boundary conditions and working towards solving the highest dimensional problem. Not all the systems belonging to the PDEs can be decoupled from each other or into smaller parts individually. The finite difference stencils S convey restrictions on the traversal possibilities .

2. Problem statement

2.1. Generalized Domains.

2.1.1. *Generalized Coordinates.* The notion of a generalized domain removes the need for a distinction between time and space coordinates x and t, respectively. To emphasize this generalized notion and accommodate a cleaner notation a generalized space-time coordinate, q, is introduced.

Definition 1. *The dimensionality of a BVP problem is given by the spatial and temporal dimensions. $d_{\mathbf{x}} \in \mathbb{N}$, noted d for confinience, denotes the spatial dimension. $d_t \in \{0,1\}$ denotes the temporal dimension. The space-time or generalized dimensionality is denoted as $d_{\mathbf{q}} := d_{\mathbf{x}} + d_t$*

$$\begin{aligned} \mathbf{x} &\in \Omega_{\mathbf{x}} \subset \mathbb{R}^{d_{\mathbf{x}}}, \\ t &\in \Omega_t \subset \mathbb{R}^{d_t} \\ \mathbf{q} &\in \Omega_{\mathbf{q}} \subset \mathbb{R}^{d_{\mathbf{q}}}. \end{aligned}$$

Definition 2. *For a d-dimensional system, the generalized coordinate $\mathbf{q}$ is a combination of spatial coordinate $\mathbf{x} = (x_1, \dots, x_d)$ and, if it is used i.e. $d_t = 1$, temporal coordinate t.*

$$\mathbf{q} := (q_1, \dots, q_{d+d_t}) := \begin{cases} (x_1, \dots, x_d) & d_t = 0, \\ (x_1, \dots, x_d, t) & d_t = 1 \end{cases} . \tag{2.1}$$

A system of space dimensionality d and time dimensionality 1 differs from a purely spatial system of space dimensionality $d+1$ and time dimensionality 0 in the assumptions made for the last coordinate. For a time dependent system the space-time domain is assumed to be decoupled as $\Omega_{\mathbf{x}} \times \Omega_t$. Furthermore Ω_t is assumed to be of the form $[0, T]$ for an end-time parameter $T \in \mathbb{R}^+$.

2.2. Generalized Domains. With higher dimensionality more complex boundary structures arise. Where a one dimensional line segments has two vertices as its boundaries, a two dimensional rectangle has four line segment as its boundaries, each of those having vertices as boundaries. Each of those vertex boundaries is shared with one other line segment. This is even further complicated by higher dimensions and more complicatedly shaped domains.

A domain Ω is assumed to be a subset of $\mathbb{R}^{d_{\mathbf{q}}}$. A BVP defines PDEs for different subsets of the domain. This subdivision results in a partition of domain Ω.

Definition 3. *Given a domain Ω and partition $\mathcal{W}$ of this domain. Each element of $\mathcal{W}$ is a **subdomain** of Ω.*

For each element ω of the partition $\mathcal{W}$ the BVP defines a PDE and a right hand side function f describing the inhomogeneous term.

The dimension of each subdomain is not necessarily equal to the domain dimension $d_{\mathbf{q}}$. Boundary regions for example have a lower dimension.

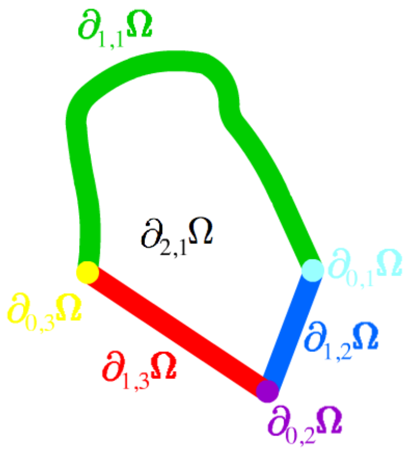
Definition 4. *For each subdomain $\omega \in \mathcal{W}$ $\#_{\mathcal{W}} : \mathcal{W} \to \{1, \ldots, \dim \Omega\} \times \mathbb{N}$ provides a unique index tuple: $\#_{\mathcal{W}}(\omega) = (k, l)$. Here $k = \dim \omega$ and l is an enumeration of ω over all k-dimensional elements of $\mathcal{W}$.*

The dimension of a subdomain, $k = \dim \omega$, can, for example, be determined by using Poincarés concept of removing a finite number of $(k-1)$-dimensional spheres to create a separation. Note that such an enumeration exists: for example the domains can be sorted using a lexicographic ordering of the lexicographic minimal point of each subdomain.

Definition 5. *Given a domain partition $\mathcal{W}$ the* ***boundary operator*** *for each subdomain ω is defined as: $\partial_{k,l}\omega := \omega'$ such that $\#_{\mathcal{N}_\omega}\omega' = (\dim \omega - k, l)$, where $\mathcal{N}_\omega$, the neighborhood of ω, is defined by all elements of $\mathcal{W}$ that are connected to ω, including ω itself. The boundary operator should not be confused with the partial derivative operator which is applied to functions.*

The definitions for dimension, subdomain, and boundary operators are formulated such that they will remain applicable to the discretized domain.

EXAMPLE 1. *2-dimensional General boundary structure*



The boundaries of this two dimensional domain provide for a boundary value problem of the following structure:

$$BVP := \begin{cases} L_{2,1}u = f_{2,1} & \forall \mathbf{q} \in \partial_{2,1}\Omega \\ \begin{cases} L_{1,1}u = f_{1,1} & \forall \mathbf{q} \in \partial_{1,1}\Omega \\ \begin{cases} L_{0,1}u = f_{0,1} & \forall \mathbf{q} \in \partial_{0,1}\Omega \end{cases} \\ \begin{cases} L_{0,3}u = f_{0,3} & \forall \mathbf{q} \in \partial_{0,3}\Omega \end{cases} \end{cases} \\ \begin{cases} L_{1,2}u = f_{1,2} & \forall \mathbf{q} \in \partial_{1,2}\Omega \\ \begin{cases} L_{0,1}u = f_{0,1} & \forall \mathbf{q} \in \partial_{0,1}\Omega \end{cases} \\ \begin{cases} L_{0,2}u = f_{0,2} & \forall \mathbf{q} \in \partial_{0,2}\Omega \end{cases} \end{cases} \\ \begin{cases} L_{1,3}u = f_{1,3} & \forall \mathbf{q} \in \partial_{1,3}\Omega \\ \begin{cases} L_{0,1}u = f_{0,1} & \forall \mathbf{q} \in \partial_{0,1}\Omega \end{cases} \\ \begin{cases} L_{0,3}u = f_{0,3} & \forall \mathbf{q} \in \partial_{0,3}\Omega \end{cases} \end{cases} \end{cases} \tag{2.2}$$

EXAMPLE 2. *d-dimensional hyperrectangular boundary structure*

A d dimensional hyperrectangle resides in a space defined with d coordinates. For each $(d-1)$- dimensional boundary one of those coordinates is fixed to either the minimal or maximal value. Resulting in $2d$ $(d-1)$- dimensional boundaries.

For the $(d-2)$- dimensional boundaries two of the d coordinates are fixed to either the minimal or maximal value. This gives $\binom{d}{d-2}$ sets of two coordinates and $4 = 2^2$ ways of fixing them to the minimal or maximal value ((min,min),(min,max),(max,min),(max,max)).

This process is further described by observing that a d-dimensional hyper rectangular domain has $2^{d-k}\binom{d}{k}$ k-dimensional boundaries.
A one dimensional example:

$$\begin{aligned}
\partial_{1,1}\Omega &= (x_{min}, x_{max}) \\
\partial_{0,1}\Omega &= \{x_{min}\} \\
\partial_{0,2}\Omega &= \{x_{max}\} \\
\Omega &= \bigcup_{k=0}^{1} \bigcup_{l=1}^{2^{1-k}\binom{1}{k}} \partial_{k,l}\Omega = [x_{min}, x_{max}]
\end{aligned}$$

A two dimensional example:

$$\begin{aligned}
\partial_{2,1}\Omega &= (x_{min}, x_{max}) \times (y_{min}, y_{max}) \\
\partial_{1,1}\Omega &= (x_{min}, x_{max}) \times \{y_{min}\} \\
\partial_{1,2}\Omega &= (x_{min}, x_{max}) \times \{y_{max}\} \\
\partial_{1,3}\Omega &= \{x_{min}\} \times (y_{min}, y_{max}) \\
\partial_{1,4}\Omega &= \{x_{max}\} \times (y_{min}, y_{max}) \\
\partial_{0,1}\Omega &= \{(x_{min}, y_{min})\} \\
\partial_{0,2}\Omega &= \{(x_{min}, y_{max})\} \\
\partial_{0,3}\Omega &= \{(x_{max}, y_{min})\} \\
\partial_{0,4}\Omega &= \{(x_{max}, y_{max})\} \\
\Omega &= \bigcup_{k=0}^{2} \bigcup_{l=1}^{2^{2-k}\binom{2}{k}} \partial_{k,l}\Omega = [x_{min}, x_{max}] \times [y_{min}, y_{max}]
\end{aligned}$$

2.3. Boundary Value Problem. A Boundary Value Problem (BVP) is defined by a partial differential equation (PDE) and the boundary conditions (BC). The PDE describes the prescribed behavior for the solution function u in the interior of the domain Ω. The BC describes the prescribed behavior for the solution function on the boundary of the domain $\partial\Omega$.

$$(BVP) = \left\{ \begin{array}{l} (PDE) \\ (BC) \end{array} \right. . \tag{2.3}$$

The boundary can be partitioned with different conditions for each boundary part.

$$(BVP) = \begin{cases} (PDE) \\ (BC)_1 \\ (BC)_2 \\ \vdots \end{cases} . \tag{2.4}$$

For time dependent problems the formulation of a special boundary condition: the initial condition (IC) is used.

$$(BVP) = \begin{cases} (PDE) \\ (IC) \\ (BC)_1 \\ (BC)_2 \\ \vdots \end{cases} . \tag{2.5}$$

For higher dimensional domains these boundaries can, in turn, be divided again into an interior and a boundary and in fact state another, lower dimensional, BVP.

These BVPs can be nested until an explicit boundary condition is used of a zero-dimensional subdomain is reached for which no differential can be defined and the BC must therefore be explicit.

A d-dimensional BVP consisting of nested boundaries can be defined as follows:

$$(BVP) = \begin{cases} (PDE)_d \\ \begin{cases} (PDE)_{d-1} \\ \ddots \begin{cases} (PDE)_0 \end{cases} \end{cases} \end{cases} . \tag{2.6}$$

Since boundary conditions can also be defined partially for boundaries, $(PDE)_k$ for a given dimension k can represent a system of PDEs. This yields a tree-like structure.

$$(BVP) = \begin{cases} (PDE)_{d,1} \\ \vdots \\ (PDE)_{d,l_d} \\ \begin{cases} (PDE)_{d-1,1} \\ \vdots \\ (PDE)_{d-1,l_{d-1}} \\ \ddots \begin{cases} (PDE)_{0,1} \\ \vdots \\ (PDE)_{0,l_0} \end{cases} \end{cases} \end{cases} . \tag{2.7}$$

Note that some of the branches of this tree may be cut short by the use of an explicit boundary condition. The PDEs are defined by a differential operator, L, and right hand function f. This gives:

$$(BVP) = \begin{cases} L_{d,1}u = f_{d,1} & \forall \mathbf{q} \in \partial_{d,1}\Omega = \mathrm{int}(\Omega) \\ \vdots & \\ L_{d,l_d}u = f_{d,l_d} & \forall \mathbf{q} \in \partial_{d,l_d}\Omega \\ \begin{cases} L_{d-1,1}u = f_{d-1,1} & \forall \mathbf{q} \in \partial_{d-1,1}\Omega \\ \vdots & \\ L_{d-1,p_{d-1}}u = f_{d-1,l_{d-1}} & \forall \mathbf{q} \in \partial_{d-1,l_{d-1}}\Omega \\ \ddots \begin{cases} L_{0,1}u = f_{0,1} & \forall \mathbf{q} \in \partial_{0,1}\Omega \\ \vdots & \\ L_{0,p_0}u = f_{0,l_0} & \forall \mathbf{q} \in \partial_{0,l_0}\Omega \end{cases} \end{cases} \end{cases} , \tag{2.8}$$

where

$$u \ : \ \Omega \to \mathbb{R} \tag{2.9}$$

$$f_{k,l} \ : \ \partial_{k,l}\Omega \to \mathbb{R} \qquad \forall 1 \leq l \leq l_k \tag{2.10}$$

To approximate u a cascading recursive traversal approach will be used to first approximate the lower dimensional problems and the climb up using those values to approximate u for the entire domain Ω.

3. Testcases

To perform experiments three test cases have been selected. They have been selected as both testing tools during the development of the methods as well as to compare the complete methods. The presence of non linear terms in x, non linear terms in u, non zero, non static boundaries and exact solution provided control mechanisms at different stages of the development process.

3.1. Testcase 1: Heat equation.

The heat equation simulates a conducting volume with insulated (zero) boundaries. Via diffusion the heat disperses and the temperature equalizes to zero along the entire volume.

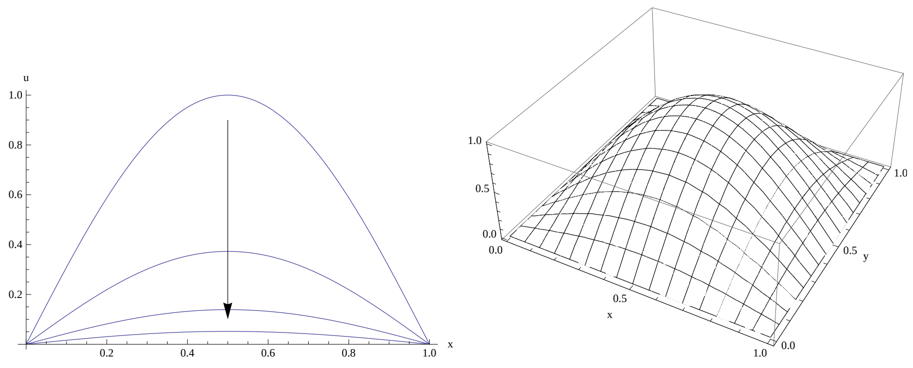


FIGURE 1. Testcase 1 : Heat equation for $d = 1, 2$.

$$
\begin{aligned}
L &:= d\frac{\partial}{\partial t} - \Delta \\
\Omega &:= [0,1]^d \\
T &:= 1 \\
u_{\text{exact}}(\mathbf{x},t) &:= e^{-t}\prod_{k=1}^{d}\sin(\pi x_k) \\
f_{IC}(\mathbf{x},t) &:\equiv u_{\text{exact}}|_{\Omega\times\{0\}} = \prod_{k=1}^{d}\sin(\pi x_k) \\
f_{BC}(\mathbf{x},t) &:\equiv u_{\text{exact}}|_{\partial\Omega} \equiv 0.
\end{aligned}
$$

Note the inclusion of dimensional parameter d as a scalar for the time derivative. It is used to create an exact solution that is equivalent throughout dimensions. If this parameter is left out, the $-t$ term in the exact solution will become $-dt$ resulting in different speeds in the propagation of the solutions for different dimensions. By equalizing these speeds the exact solution to the d-dimensional problem restricted to the domain of any lower dimensional problem is equivalent to the exact solution of that lower dimensional problem.

3.2. Testcase 2: Implosion equation. The implosion equation features origin oriented advection that compresses the starting volume into a thin needlelike solution. The advection increases linearly with the distance from the origin. The boundaries are zero at infinity but are already close to zero at much smaller distance to the origin.

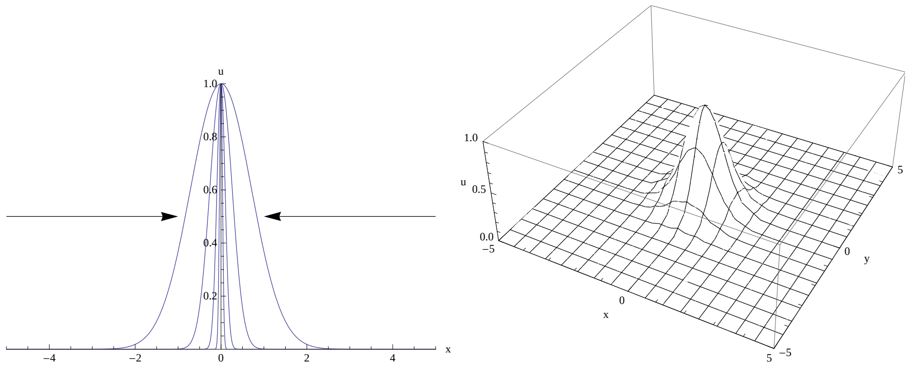


FIGURE 2. Testcase 2: Implosion equation for $d = 1, 2$.

$$
\begin{aligned}
L &:= \frac{\partial}{\partial t} - \mathbf{x} \cdot \nabla \\
\Omega &:= [-5, 5]^d \\
T &:= 5 \\
u_{\text{exact}}(\mathbf{x}, t) &:= e^{-e^{2t} ||\mathbf{x}||_2^2} \\
f_{IC}(\mathbf{x}, t) &:\equiv u_{\text{exact}}|_{\Omega \times \{0\}} = e^{-||\mathbf{x}||_2^2} \\
f_{BC}(\mathbf{x}, t) &:\equiv u_{\text{exact}}|_{\partial\Omega} \approx 0.
\end{aligned}
$$

Contrary to the other test cases no dimensional parameter is needed such that the exact solution to the d-dimensional problem restricted to the domain of any lower dimensional problem is equivalent to the exact solution of that lower dimensional problem.

3.3. Testcase 3: Burgers equation. Burgers equation features a wave propelled by a non linear advection term coupled with diffusion regulated by parameter μ.

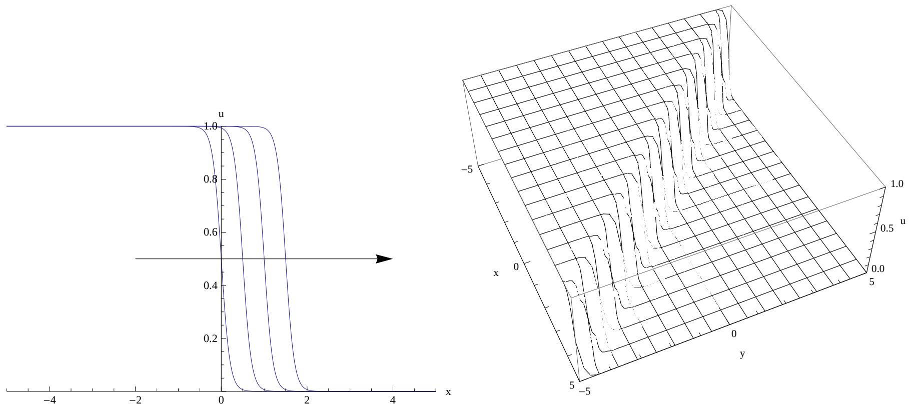


FIGURE 3. Testcase 3: Burgers equation for $d = 1, 2$.

$$
\begin{aligned}
L &:= d\frac{\partial}{\partial t} + u(\mathbf{1} \cdot \nabla) - \mu\Delta \\
\Omega &:= [-5, 5]^d \\
T &:= 5 \\
u_{\text{exact}}(\mathbf{x}, t) &:= \frac{1}{1 + e^{-\frac{1}{4\mu} + \sum_{k=1}^{d} \frac{x_k}{2\mu}}} \\
f_{IC}(\mathbf{x}, t) &:\equiv u_{\text{exact}}|_{\Omega \times \{0\}} \\
f_{BC}(\mathbf{x}, t) &:\equiv u_{\text{exact}}|_{\partial\Omega}.
\end{aligned}
$$

Note again the inclusion of dimensional parameter d as a scalar for the time derivative. It is used to create an exact solution that is equivalent throughout dimensions. If this parameter is left out the $-\frac{1}{4\mu}$ term in the exact solution will become $-\frac{d}{4\mu}$ resulting in different wave speeds in the solutions for different dimensions. By equalizing the wave speeds the exact solution to the d-dimensional problem restricted to the domain of any lower dimensional problem is equivalent to the exact solution of that lower dimensional problem.

4. Differential operators

4.1. Linear differential operators. A linear differential operator L, which can be defined using a multivariate polynomial, P_L, in variables $\partial_{q_1}, \dots, \partial_{q_d}$ where ∂_{q_k} is defined as $\partial_{q_k} := \frac{\partial}{\partial q_k}$, consequently $\partial_{q_k}^j = \frac{\partial^j}{\partial q_k^j}$.

P_L can be represented by coefficients s_i, a scalar, and $\mathbf{j}_i$, a vector containing exponent coefficients, for each term indexed by i:

$$L := P_L[\partial_{q_1}, \dots, \partial_{q_d}] := \sum_i s_i \bigcirc_{k=1}^{d} \partial_{\mathbf{q}_k}^{(\mathbf{j}_i)_k}, \tag{4.1}$$

where $\bigcirc$ indicates the repeated function composition contrasting the repeated multiplication operator, $\prod$. L can now be denoted using a stencil representation:

$$\bigcup_i \{(s_i, \mathbf{j}_i)\}. \tag{4.2}$$

I refrain from storing the coefficients in vectors or matrices because I want to leave the option for fractional exponents open. At first not for the differential operator, though fractional derivates do arise in certain field, but especially for the, yet to be defined, finite difference operator which will adhere to an analogous stencil notation. This stencil, or sparse, notation will also be better suited for cases where the label options are numerous.

Example 3. *Consider the equations:*

$$u = \alpha u_x; \tag{4.3}$$

$$v = \beta v_x + \gamma v_{yy}, \tag{4.4}$$

their corresponding differential operators:

$$L_u = \mathrm{id} - \alpha \frac{\partial}{\partial x}; \tag{4.5}$$

$$L_v = \mathrm{id} - \beta \frac{\partial}{\partial x} - \gamma \frac{\partial^2}{\partial y^2}, \tag{4.6}$$

where id *can be denoted as a scalar 1 in many cases but* $\mathrm{id}(\cdot)^2$ *is an obvious exception (i.e.* $\mathrm{id}^2(u) = \mathrm{id}(u) = 1u$ *but* $,\mathrm{id}(u)^2 = u^2$*). Their stencil forms are given by:*

$$L_u = \{(1,\mathbf{0}),(-\alpha,\mathbf{e}_1)\}; \tag{4.7}$$
$$L_v = \{(1,\mathbf{0}),(-\beta,\mathbf{e}_1),(-\gamma,2\mathbf{e}_2\}, \tag{4.8}$$

An overview for the differential operators:

	Differential operators
Solution	u
Atom	∂_{q_k}
Atom	id
Atom	$\mathcal{D}_{\mathbf{v}}$
Molecule	L

The stencil operations of scaling, addition, subtraction and multiplication are defined in Appendix 1.

4.2. Special linear differential operators. Two operators deserve special mention: the Laplace operator $\Delta := \sum_{k=1}^{d} \frac{\partial^2}{\partial q_d^2}$ and the Gradient operator $\nabla := \begin{pmatrix} \frac{\partial}{\partial q_1} \\ \vdots \\ \frac{\partial}{\partial q_d} \end{pmatrix}$.

Note that these are, by default defined over the spatial coordinates $x_1,\ldots,x_d$ not the time coordinate t. A subscript parameter can be used to distinguish other cases:

$$\Delta := \Delta_{\mathbf{x}} := \sum_{k=1}^{d} \frac{\partial^2}{\partial x_k^2}, \qquad \Delta_{\mathbf{q}} := \sum_{k=1}^{d+1} \frac{\partial^2}{\partial q_k^2}. \tag{4.9}$$

And for a general coordinate vector $\mathbf{a}$:

$$\Delta_{\mathbf{a}} := \sum_{k=1}^{|\mathbf{a}|} \frac{\partial^2}{\partial a_k^2}. \tag{4.10}$$

The directional derivative $\mathcal{D}_{\mathbf{v}} := \mathbf{v}\cdot\nabla$ is defined for the derivative in the direction of $\mathbf{v}$.

4.3. Non linear differential operators. The stencil concept can be extended to include a vastly larger set of differential operators by stating that s needs not be a scalar but can also be a function: $s(\cdot):\Omega\to\mathbb{R}$. This results in a differential operator that is still linear in u but non linear in $\mathbf{x},t$. Some examples:

$$u(x) = \cos(x)u(x)+\sin(x)u_x(x); \tag{4.11}$$
$$v^2 = v_x, \tag{4.12}$$

their corresponding differential operators:

$$L_u = \mathrm{id} - \cos(\cdot)\mathrm{id} - \sin(\cdot)\frac{\partial}{\partial x}; \tag{4.13}$$

$$= (1 - \cos(\cdot))\mathrm{id} - \sin(\cdot)\frac{\partial}{\partial x}; \tag{4.14}$$

$$L_v = \mathrm{id}(\cdot)^2 - \frac{\partial}{\partial x} \tag{4.15}$$

$$= v(\cdot)\mathrm{id} - \frac{\partial}{\partial x} \tag{4.16}$$

and their stencil forms:

$$L_u = \{(1 - \cos(\cdot), \mathbf{0}), (-\sin(\cdot), 2\mathbf{e}_1)\}; \tag{4.17}$$

$$L_v = \{(v(\cdot), \mathbf{0}), (-1, \mathbf{e}_1\}, \tag{4.18}$$

The last example of L_v is already a set up for finite difference discretization. It exemplifies a case where L is non linear in solution u. When solving a PDE for v, $v(\cdot)$ is of course not yet available. A previous time step or iteration of v, v^{n-1} can be used as an approximation in an effort to solve the quadratic equation:

$$Lv^n = \{(v^{n-1}(\cdot), \mathbf{0}), (-1, \mathbf{e}_1\}v^n = f \tag{4.19}$$

The $(\cdot)$ notation can also be dropped when it is clear which variables are scalars and which are functions. As one is to expect with functions as famous as cos and sin.

CHAPTER 2

Generalized d-dimensional FDM

1. Generalized Domain Discretization

The challenge ahead lies in defining a method, and an accompanying notation, to provide proper index administration for complex structures that can arise in the discretization of domains. These domains are not restricted to one, two or three dimensions but can be of any dimension d. Furthermore they are not limited to uniform grid but also provide administration for non conforming grids such as might arise from h-refinement at a later stage. Although this latter property is not used in this thesis I found that defining a rigorous foundation, even for the well known uniform grids, allowed easier handling of higher dimensional grids

To perform the manipulation required I introduce a new definition for a domain discretization on which the finite difference can be applied:

Definition 6. *A **discretization** of domain Ω is defined using a tuple $W = (\mathcal{W}, \psi, \mathcal{J}_i, \Delta_i)$.*

$\mathcal{W}$ is a partition of a set $\mathcal{I}$ such that $\mathcal{I} \subset \mathbb{N}$. This set is thereforegiven by:

$$\mathcal{I} := \bigcup_{\omega \in \mathcal{W}} \omega. \tag{1.1}$$

*$\mathcal{I}$ is called the **index set**. It contains the indices for each node used in the discretization of the domain.*

Each index, i, is mapped to a node vector, $\mathbf{q}_i$, in the continuous domain Ω by the node function ψ. The image of ψ is a discrete subset of Ω given by the node set $\mathcal{Q}$.

$$\psi \quad : \quad \mathcal{I} \rightarrow \mathcal{Q} \tag{1.2}$$

*ψ gives the location of a node given its index number. Besides the location of the nodes a notion of **spatial-temporal relation** is needed. This **neighborhood** or **adjacency** to form a grid is defined using a set of adjacency labels $\mathcal{J}_i$ and an adjacency operator Δ_i for each node index i .*

*The **adjacency operator**,Δ_i, defines a **neighborhood** or **adjacency set** for each node index i using the **adjacency label set** $\mathcal{J}_i$. The adjacency operator maps an adjacency label to an index offset:*

$$\Delta_i \quad : \quad (\mathcal{J}_i \cup \{0\}) \rightarrow \mathcal{N}_i \cup \{i\} \qquad \forall i \in \mathcal{I} \tag{1.3}$$

$$\mathcal{N}_i \quad := \quad \{i + \Delta j | j \in \mathcal{J}_i\} \subset \mathcal{I} \tag{1.4}$$

For each adjacency operator it must hold that $\Delta_i 0 = 0$. Such that $i + \Delta_i 0 = i + 0 = i$. In other words: the zero label always yields a zero offset and thus ends up where it started.

(1.5) $$\Delta_i j$$

The **adjacency operator**, Δ_i, indicates what offset should be added to arrive at the index of the neighbor label by j.

(1.6) $$\mathcal{J}_i$$

The **adjacency label set**, $\mathcal{J}_i$, indicates what the labels of the surrounding neighbors are.

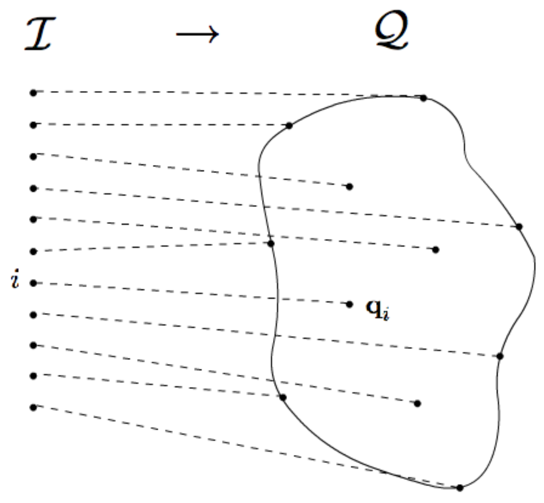


FIGURE 1. ψ maps the index set $\mathcal{I}$ to the node set $\mathcal{Q}$.

EXAMPLE 4. ***Left and right neighbors***

Given the nodes indexed from 1 to 5.

When we look at node 3 the label set could be defined as $\mathcal{J}_3 = \{left, right\}$. The adjacent operator will map those labels to their corresponding offsets:

$$\Delta_3(right) = 1$$
$$\Delta_3(left) = -1$$

Such that we find neighbors $i + \Delta_3(right) = 4$ and $i + \Delta_3(left) = 2$.

Example 4 might seem pretty straightforward but with higher dimensional domains, their boundaries, and topologically more intricate domains or node relations proper administration of the neighborhoods is key to unraveling the structures that can be beneficial to efficient computations.

Consider for example the adjacency label sets for nodes 1 and 5, $\mathcal{J}_1, \mathcal{J}_5$. These will include only the right respectively left label. For higher dimensions these differences in adjacent sets become more complicated and will influence the structure of the linear systems derived from the discretizations.

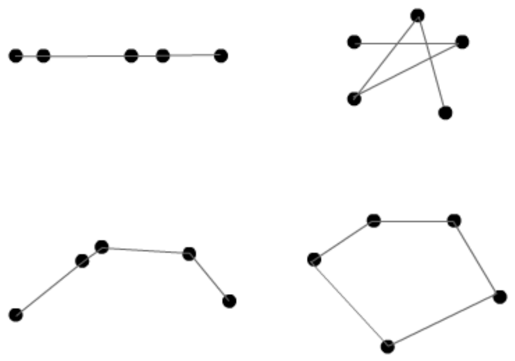
The index set of example 4 is given by $\mathcal{I} = \{1, 2, 3, 4, 5\}$ and the node count is $m = 5$. Here the **node count** m is the number of nodes used in the discretization of Ω

$$m \quad := \quad |\mathcal{I}| = |\mathcal{Q}| \tag{1.7}$$

The node set $\mathcal{Q}$ must have the same number of elements but without the definition of ψ its elements are undefined. The image and use of left and right labels suggest a straight one-dimensional domain, but this is not necessarily so. The scaling could be different but the domain could be shaped completely different as well.

EXAMPLE 5. ***Left and right neighbors: exotic cases***

Given the nodes indexed from 1 to 5 already used in example 4 the introduction of ψ could yield totally different node sets $\mathcal{Q}$:



Ranging from differently spaced nodes to curves, crossing 'edges' and even cyclical structures if an adjacency label set of $\mathcal{J} = \{right, left\}$ is used for all nodes.

To analyze the options arising from example 5 and various others the relative position of neighboring nodes i given by:

$$\mathbf{r}_{i,j} = \mathbf{q}_{i+\Delta j} - \mathbf{q}_i \tag{1.8}$$

is needed. Besides their relative position , distances to the neighboring nodes are defined. This is not a straightforward Euclidian distance but a pseudo norm that retains some information on the direction. This information is later needed for the finite difference approximations.

$h_j : \mathcal{I} \rightarrow \mathbb{R}$ is a sign preserving pseudo norm that maps the distance to neighboring nodes:

$$h_j(i) = \text{sign}(\langle |\mathbf{r}_{i,j}|, \mathbf{r}_{i,j} \rangle)||\mathbf{r}_{i,j}|| \tag{1.9}$$

2. Boundaries

To introduce the definitions for dimensionality and boundaries for the discretized domains, I begin by defining the index and node equivalents to the domain and subdomain defined in 2.2 for the index set $\mathcal{I}$ and the node set $\mathcal{Q}$:

$$\mathcal{I}_{\Omega'} := \mathcal{I} \bigcap \psi^{-1}\left(\Omega'|_{\mathcal{Q}}\right) \tag{2.1}$$

$$\mathcal{Q}_{\Omega'} := \mathcal{Q} \bigcap \Omega' \tag{2.2}$$

To implement the definitions for boundary operators only a revised definition of connectivity is needed. The adjacency operator is a very good candidate but note that, while the boundary operators can now be applied to an index subdomain $\mathcal{I}_\omega$ or a node subdomain, subdomain $\mathcal{Q}_\omega$ in general it does not hold that:

$$\partial_{k,l}\mathcal{I}_\omega = \mathcal{I}_{\partial_{k,l}\omega} \tag{2.3}$$

$$\partial_{k,l}\mathcal{Q}_\omega = \mathcal{Q}_{\partial_{k,l}\omega} \tag{2.4}$$

Properties that might be desirable for the proper representation of a BVP but are not explored in this thesis.

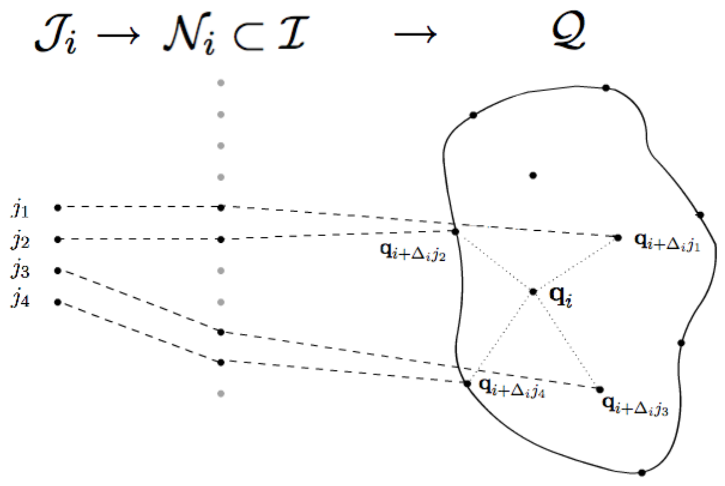

$\mathcal{J}_i \rightarrow \mathcal{N}_i \subset \mathcal{I} \rightarrow \mathcal{Q}$
j_1
j_2
j_3
j_4
$\mathbf{q}_{i+\Delta_i j_2}$
$\mathbf{q}_{i+\Delta_i j_1}$
$\mathbf{q}_i$
$\mathbf{q}_{i+\Delta_i j_4}$
$\mathbf{q}_{i+\Delta_i j_3}$

FIGURE 2. Δ_i maps the adjacency label set $\mathcal{J}_i = \{j_1, j_2, j_3, j_4\}$ to the adjacency set $\mathcal{N}_i$.

DEFINITION 7. *A discretization $W = (\mathcal{I}, \psi, \mathcal{J}_i, \Delta_i)$ is called* ***regular*** *if for each subdomain $\omega \in \mathcal{W}$ it holds that there exists a set $\mathcal{J}$ and a function Δ such that $\mathcal{J}_i \equiv \mathcal{J}$ and $\Delta_i \equiv \Delta$ for all $i \in \omega$. W is called a* ***lattice*** *when for each subdomain ω the relative neighbor positions also coincide: $R_i \equiv R$ for all $i \in \omega$. The* ***degree*** *of a regular discretization $W = (\mathcal{I}, \psi, \mathcal{J}_i, \Delta_i)$ is given by the degree of the interior nodes $|\mathcal{J}|$.*

Regularity and, even more so, lattices will be useful because they provide a more structured linear system that is easier to analyze and solve.

Definition 8. *A discretization $W = (\mathcal{I}, \psi, \mathcal{J}_i, \Delta_i)$ is called* ***transitive*** *if for each $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$ $\Delta_i(j_1 + j_2) = \Delta_i j_1 + \Delta_{i+\Delta_i j_1} j_2$ holds.*

Transitivity will provide approximations for higher order differentials. It represents the ability to describe the neighbor of a neighbor of a node directly relative to that node. (i.e. the 'northern' neighbor of my 'western' neighbor is my 'northwestern' neighbor).

Definition 9. *A discretization is called* ***spanning*** *if for each node i in each subdomain $\partial_{k,l}\mathcal{I}$ there exists a subset $\{j_1, \ldots, j_k\} \subseteq \mathcal{J}_i$ such that $\mathbf{r}_{i,j_1}, \ldots \mathbf{r}_{i,j_k}$ spans $\partial_{k,l}\Omega$.*

The spanning property ensures that derivatives in any direction can be approximated using this domain discretization.

In further chapters discretization properties favorable to a finite difference approach will become apparent. For one dimensional discretizations some statements can already be made:

Lemma 1. *Any strictly one dimensional connected discretization $\mathcal{W}$ of degree 2 can be indexed and labeled such that it is transitive.*

Choose the adjacency label set as $\mathcal{J}_i \equiv \{1, -1\}$.

If $\mathcal{W}$ is not cyclical then the adjacency operator can be defined as

$$\Delta_i \equiv id \tag{2.5}$$

Let the nodes be sorted for example left to right and numbered accordingly. Now label their predecessor by -1 and their successor by 1 .

Transitivity is shown by $\Delta_i j_1 + \Delta_{i+\Delta_i j_1} j_2 = j_1 + j_2 = \Delta_i(j_1 + j_2)$ $\square$

If $\mathcal{W}$ is cyclical then the adjacency operator can be defined as

$$\Delta_i(j) = (i + j) \ mod \ I - i \tag{2.6}$$

Transitivity is shown by

$$\begin{aligned}
& \Delta_i j_1 + \Delta_{i+\Delta_i j_1} j_2 \\
= \ & (i + j_1) \ mod \ I - i + (i + \Delta_i j_1 + j_2) \ mod \ I - (i + \Delta_i j_1) \\
= \ & -i + (i + \Delta_i j_1 + j_2) \ mod \ I \\
= \ & -i + (i + (i + j_1) \ mod \ I - i + j_2) \ mod \ I \\
= \ & -i + ((i + j_1) \ mod \ I + j_2) \ mod \ I \\
= \ & -i + ((i + j_1) \ mod \ I + j_2 \ mod \ I) \ mod \ I \\
= \ & (i + j_1 + j_2) \ mod \ I - i \\
= \ & \Delta_i(j_1 + j_2) \quad \square
\end{aligned}$$

Anti-symmetry is shown by

$$\begin{aligned}
& -\Delta_{i+\Delta_i j}(-j) \\
= & -(i + \Delta_i j - j) \ mod \ I + (i + \Delta_i j) \\
= & -(i + (i+j) \ mod \ I - i - j) \ mod \ I + (i + (i+j) \ mod \ I - i) \\
= & -((i+j) \ mod \ I - j) \ mod \ I + ((i+j) \ mod \ I) \\
= & -i \ mod \ I + (i+j) \ mod \ I \\
= & (i+j) \ mod \ I - i \\
= & \Delta_i(j) \quad \square
\end{aligned}$$

For most one dimensional cases a domain discretization can be made transitive and anti-symmetric providing favorable properties. This is, in general, not the case for discretizations of higher dimensional domains.

3. Function Discretization

A function $f : \Omega \to \mathbb{R}$ can be discretized using a vector $\mathbf{f}$ containing nodal values such that:

$$f_i \ := \ f(\mathbf{q}_i) \qquad i \in \mathcal{I} \tag{3.1}$$

and a notational extension to create vectors containing all nodal values for a given subset $\mathcal{A}$ of the index set $\mathcal{I}$:

$$\mathbf{f}^{\mathcal{A}} \ := \ \underset{i \in \mathcal{A}}{\boxed{\downarrow}} \ f(\mathbf{q}_i) = \begin{pmatrix} \vdots \\ f(\mathbf{q}_i) \\ \vdots \end{pmatrix}, \tag{3.2}$$

where the index values are in ascending order.

4. Finite difference approximation

4.1. Finite difference operators. This section is focussed on the conversion, or approximation, from a linear differential operator, L, to a linear finite difference operator, S. First an implicit, then an explicit definition in the form of a stencil representation will be proposed.

$$L \rightsquigarrow S. \tag{4.1}$$

The approach will be based on handling the atomic parts of the operators first and transforming those:

$$\frac{\partial}{\partial q_k} \rightsquigarrow \sigma_{(\cdot)_{q_k}}. \tag{4.2}$$

Given an atomic differential operator, $\frac{\partial}{\partial q_k}$, an atomic finite difference approximation stencil, $\sigma_{(\cdot)_{q_k}}$, is created. There can be several distinct approximations, $\sigma_{(\cdot)q_k}$, of $\frac{\partial}{\partial q_k}$, where $(\cdot)$ implies the freedom to choose from different approximation methods e.g. Forward, Backward, Central or many others. The $(\cdot)q_k$ implies that no choice has been made but the choice should approximate $\frac{\partial}{\partial q_k}$.

$$\frac{\partial}{\partial q_k}|_{\mathcal{Q}} u \approx \sigma_{(\cdot)q_k}\mathbf{u}, \tag{4.3}$$

where $\mathcal{Q}$ is the node set, $\mathcal{I}$ the node index set, and $\mathbf{u}$ the discretization of the solution function as defined in (3.2).

The approach then recombines the atoms using the multivariate polynomial, P_L, that can be used to describe the general linear differential operator, L:

$$L := P_L[\frac{\partial}{\partial q_1}, \ldots, \frac{\partial}{\partial q_d}] \rightsquigarrow P_L[\sigma_{(\cdot)q_1}, \ldots, \sigma_{(\cdot)q_d}] := S. \tag{4.4}$$

These operators can then be applied to the (discretized) solution function to approximate the partial differential equation (PDE) using a finite difference equation (FDE).

$$\begin{array}{ccc} \text{(PDE)} & & \text{(FDE)} \\ Lu|_{\mathcal{Q}} = f|_{\mathcal{Q}} & \rightsquigarrow & S\mathbf{u}^{\mathcal{I}} = \mathbf{f}^{\mathcal{I}} \end{array}. \tag{4.5}$$

4.2. Atomic finite difference operators.

4.2.1. *Molecules, atoms and subatoms.* The term *atomic* is taken from the physical/chemical nomenclature. The complete differential operator is viewed as the *molecule.* For example $L = \frac{\partial}{\partial x} + \frac{\partial}{\partial y}$. The individual partial derivatives $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ are *atoms.* Just as with conventional molecules, atoms can also be molecules by themselves e.g. $L = \frac{\partial}{\partial x}$.

Their analogous finite difference operators adhere to the same structure: the molecule approximating the differential molecule and the atom approximating the differential atom. This nomenclature can be further extended by the concept of subatomic particles. These, in general, cannot function as atoms or molecules by themselves but in combinations form the building blocks for the atoms. The stencil representations of these operators are depicted below:

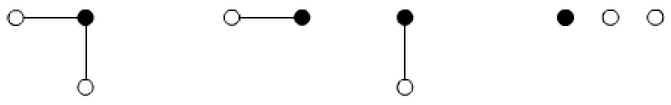


FIGURE 3. From left to right: the molecule representing a backward in space approximation of $\frac{\partial}{\partial x} + \frac{\partial}{\partial y}$; the atom representing the backward approximation of $\frac{\partial}{\partial x}$; the atom representing the backward approximation of $\frac{\partial}{\partial y}$ and the individual sub atoms of the finite difference operators.

4.2.2. *One dimensional case.* The basic concept of finite difference approximation of a differential of a one dimensional function f is given by:

$$f'(q) \approx \frac{f(q+h)-f(q)}{h}, \tag{4.6}$$

where h is the finite difference parameter. Instead of the derivative of f it gives the derivative of the linear approximation of f between q and $q+h$ which is shown graphically in Figure 4.

For a one dimensional grid as shown in Lemma 1 there is a straightforward ordering of nodes $\Delta_i = \text{id}$ and thus $\mathbf{f}_{i+\Delta_i 1} = \mathbf{f}_{i+1}$ and

$$\frac{f(q_i+h)-f(q_i)}{h} \tag{4.7}$$

$$= \frac{\mathbf{f}_{i+1}-\mathbf{f}_i}{q_{i+1}-q_i} \tag{4.8}$$

$$= \frac{\mathbf{f}_{i+\Delta_i 1}-\mathbf{f}_i}{q_{i+\Delta_i 1}-q_i} \tag{4.9}$$

$$= \frac{\mathbf{f}_{i+\Delta_i 1}-\mathbf{f}_i}{r_{i,1}} \tag{4.10}$$

$$= \frac{\mathbf{f}_{i+\Delta_i 1}-\mathbf{f}_i}{h_1(i)}. \tag{4.11}$$

By defining the sub atomic finite difference operator ς^j as

$$\left(\varsigma^j \mathbf{f}\right)_i := \mathbf{f}_{i+\Delta_i j} \tag{4.12}$$

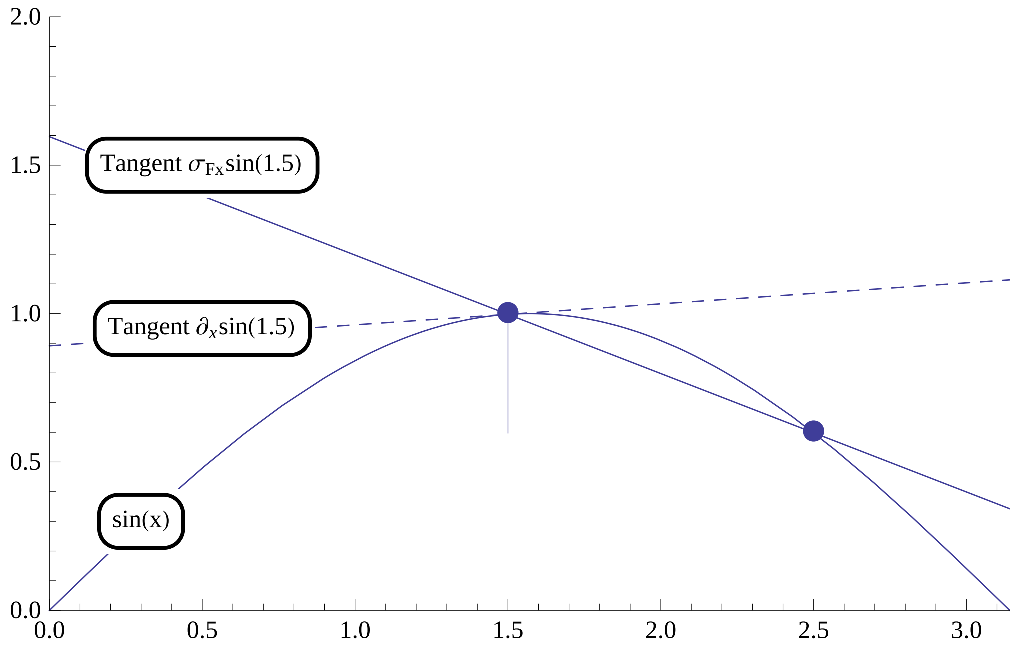


FIGURE 4. The finite difference approximation of $\frac{\partial}{\partial x}\sin(1.5)$ using nodes $q_i = 1.5$ and $q_{i+1} = 1.5 + h = 2.5$.

such that (4.11) can be written as

$$= \left(\frac{\varsigma^1 - \varsigma^0}{h_i(1)}\mathbf{f}\right)_i \approx f'(q_i). \tag{4.13}$$

ς^j

The sub atomic finite difference operator ς^j is a shifting operator that shifts nodes with i to their j-th neighbor. The shift is given by $\Delta_i j$ and the index of the neighbor is thereforegiven by: $i + \Delta_i j$.

Each sub atom is based on a neighboring node. The relative position is indicated by the *label* $j \in \mathcal{J}_i$. The definition for ς^j is implicit for now but will be made explicit when the matrix representations are handled. Note that, as one might expect of a zero exponent:

$$\left(\varsigma^0 \mathbf{f}\right)_i := \mathbf{f}_{i+0} \Rightarrow \varsigma^0 = \mathrm{id}|_{\mathcal{Q}}. \tag{4.14}$$

When it comes to approximating differential operators, ς^0 is the only subatomic operator that can be used by itself. (Excluding exotic cases where one would have a PDE of the form $\frac{\partial f}{\partial x}(x) = f(x+h)$.

From these sub atomic operators we now construct *atomic* operators σ that will approximate first order derivatives:

$$\sigma^j := \frac{\varsigma^j - \varsigma^0}{h_j(i)}. \tag{4.15}$$

For $j = 1$ it holds that:

$$\left(\sigma^1 \mathbf{f}\right)_i \approx f'(q_i) = \frac{\partial f}{\partial q}(q_i) \tag{4.16}$$

$$\Rightarrow \quad \sigma^1 \approx \frac{\partial}{\partial q}|_{\mathcal{Q}}. \tag{4.17}$$

Analogously to using the 1 label, which points forward to i's successor $i+1$ we can also use the -1 label which points backwards to i's predecessor $i-1$:

$$\sigma^{-1} \quad := \quad \frac{\varsigma^{-1} - \varsigma^0}{h_{-1}(i)} \tag{4.18}$$

$$\tag{4.19}$$

For which

$$\begin{aligned}
\left(\sigma^{-1}\mathbf{f}\right)_i &= \left(\frac{\varsigma^{-1} - \varsigma^0}{h_{-1}(i)}\mathbf{f}\right)_i && (4.20)\\
&= \frac{\mathbf{f}_{i+\Delta_i(-1)} - \mathbf{f}_i}{h_{-1}(i)} && (4.21)\\
&= \frac{\mathbf{f}_{i+\Delta_i(-1)} - \mathbf{f}_i}{r_{i,-1}} && (4.22)\\
&= \frac{\mathbf{f}_{i+\Delta_i(-1)} - \mathbf{f}_i}{q_{i+\Delta_i(-1)} - q_i} && (4.23)\\
&= \frac{\mathbf{f}_{i-1} - \mathbf{f}_i}{q_{i-1} - q_i} && (4.24)\\
&= \frac{f(q_i) - f(q_i - h)}{h} && (4.25)\\
&\approx f'(q_i). && (4.26)
\end{aligned}$$

From which it follows that also $\sigma^{-1} \approx \frac{\partial}{\partial q}|_{\mathcal{Q}}$.

σ^1 and σ^{-1} are two distinct approximations of the derivative operator they are named the *forward (F)* respectively *backward (B)* finite difference operators.

$$\text{Forward:} \qquad \sigma_{Fq} \;:=\; \sigma^1, \tag{4.27}$$

$$\text{Backward:} \qquad \sigma_{Bq} \;:=\; \sigma^{-1}. \tag{4.28}$$

The next step is to combine atoms into molecules. This waym better approximations and approximations of higher order derivatives can be constructed. Combining the forward and backward operator into a weighted average yields the *theta-operator (Θ)*:

$$\text{Theta:} \qquad \sigma_{\Theta x}(\theta) := \theta\sigma_{Fx} + (1-\theta)\sigma_{Bx} \tag{4.29}$$

with as a special case for $\theta = \frac{1}{2}$: the *central* (*C*) operator:

$$\text{(4.30)} \qquad \text{Central:} \qquad \sigma_{Cx} := \sigma_{\Theta x}\left(\frac{1}{2}\right) = \frac{\sigma_{Fx} + \sigma_{Bx}}{2}.$$

The central approximation is a second order finite difference approximation. This should not be confused with a finite difference approximation of a second order derivative. Higher order approximations are possible by combining more atoms and using more surrounding nodes. Vast families of methods arise from these combinations.

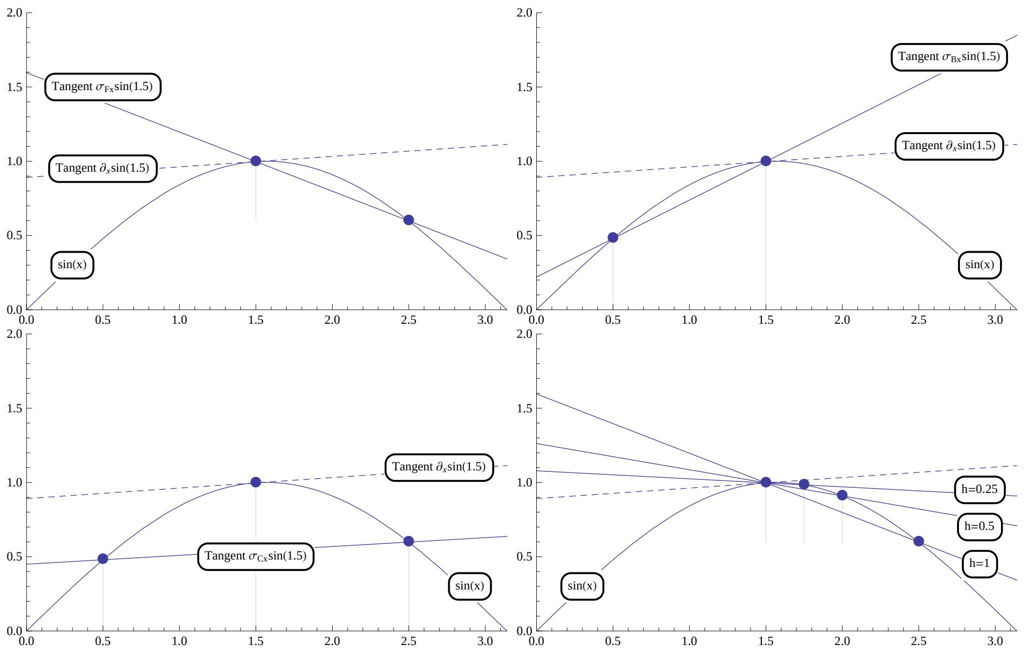


FIGURE 5. The first three images shows the difference in approximations of $\frac{\partial}{\partial x}\sin(1.5)$ for $h_x = 1$ by the forward, backward and central approximation. The last shows the difference for the forward approximation at different values of $h_x \equiv h_1 \equiv h_{-1} \equiv 1, 0.5, 0.25$ constant for all i.

As already demonstrated for the one dimensional case, solutions need not be unique. This is indicated by the $(\cdot)$ notation in $\sigma_{(\cdot)q_k}$. This is a non explicitly defined operator that approximates $\frac{\partial}{\partial q_k}$.

Each occurrence of $\frac{\partial}{\partial q_k}$ in L can be approximated by a different choice of stencil. This also applies for the n $\frac{\partial}{\partial q_k}$-terms in $\frac{\partial^n}{\partial q_k^n}$. Occurrences in special cases or higher order derivatives can benefit greatly from smart choices. Examples of this are given in section 4.6.

4.2.3. *Higher dimensional case.* An extension to higher dimensions runs into the following problem: in one dimension there are only two directions: forward and backward (or left and right). These directions can be indicated by the base vectors of one dimensional space $e_1 = 1$ and $-e_1 = -1$. Any relative position between two nodes i and j is given by $r_{i,j} := x_i - x_j$ and is always a scalar multiple of both directions. The spaces is always spanned by $r_{i,j}$ for any $j \neq 0$. So a first order derivative in one dimension can always be approximated by using a node and any of its neighbors. For higher dimensions this is not always the case: whether the relative positions of the neighborhood of a node can span the space becomes a prerequisite or a restriction to the approximation of derivatives.

The d dimensional generalized version of finite difference approximation σ^j approximates the first order derivative in the direction of $r_{i,j}$:

$$\begin{aligned}
\left(\sigma^j \mathbf{f}\right)_i &:= \frac{\mathbf{f}_{q+\Delta_i j} - \mathbf{f}_i}{h_j(i)} && (4.31)\\
&= \frac{f(\mathbf{q}_{i+\Delta_i(j)}) - f(\mathbf{q}_i)}{\operatorname{sign}(\langle |\mathbf{r}_{i,j}|, \mathbf{r}_{i,j}\rangle)||\mathbf{r}_{i,j}||} && (4.32)\\
&= \frac{f(\mathbf{q}_i + \mathbf{r}_{i,j}) - f(\mathbf{q}_i)}{\operatorname{sign}(\langle |\mathbf{r}_{i,j}|, \mathbf{r}_{i,j}\rangle)||\mathbf{r}_{i,j}||} && (4.33)\\
&\approx \mathcal{D}_{\mathbf{r}_{i,j}} f(\mathbf{q}_i). && (4.34)
\end{aligned}$$

σ^j approximates the directional derivative $\mathcal{D}_{\mathbf{r}_{i,j}}$ only locally because $\mathbf{r}_{i,j}$ still depends on node index i. For a lattice $R_i \equiv R$ so $\mathbf{r}_{i,j} \equiv \mathbf{r}_j$ does hold such that $D^j \approx \mathcal{D}_{\mathbf{r}_j}|_{\mathcal{Q}}$. This follows from:

$$\mathcal{D}_{\mathbf{v}} f(\mathbf{q}) = \lim_{h \to 0} \frac{f(\mathbf{q} + h\mathbf{v}) - f(\mathbf{q})}{h}. \tag{4.35}$$

Using just σ^j only approximations of derivatives in the direction of $\mathbf{r}_{i,j}$ can be made for labels $j \in \mathcal{J}$. To extend these directions we define

$$\sigma_{\boldsymbol{\alpha}} := \sum_{j \in \mathcal{J}} \alpha_j \sigma^j. \tag{4.36}$$

To transform a given directional derivative $\mathcal{D}_{\mathbf{v}}$ into an approximating $\sigma_{\boldsymbol{\alpha}}$ operator the system needs to be solved:

$$R_i \boldsymbol{\alpha} = \frac{\mathbf{v}}{||\mathbf{v}||}, \tag{4.37}$$

where R_i is a matrix with columns $r_{i,j}$ scaled by their sign preserved length:

$$R_i = \underset{j \in \mathcal{J}}{\boxed{\rightarrow}} \frac{\mathbf{r}_{ij}}{h_j(i)}. \tag{4.38}$$

Whether a solution exists is equivalent to the inclusion of $\mathbf{v}$ in the span of relative locations of the neighboring nodes. Definition 9 provides a spanning classification for grids that posses such a property for each node and each vector $\mathbf{v}$.

For solution $\boldsymbol{\alpha}$ of (4.37) it follows that:

$$\sigma_{\boldsymbol{\alpha}} := \sum_{j \in \mathcal{J}} \alpha_j \sigma^j \approx \mathcal{D}_{\mathbf{v}}. \tag{4.39}$$

with $\sum_{j\in\mathcal{J}} \alpha_j = 1$ and $\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{J}|}$.

For anti-symmetric discretizations a subset of the neighborhood suffices. When from each anti-symmetric pair $j = -j'$ one is chosen to create neighborhood $\mathcal{J}'$ it suffices to use

$$\sigma_{\boldsymbol{\alpha}} := \sum_{j\in\mathcal{J}'} |\alpha_j| \left\{ \begin{array}{cc} \sigma^{-j} & \alpha_j < 0 \\ \sigma^{j} & \alpha_j > 0 \end{array} \right. \approx \mathcal{D} \text{ for} \sum_{j\in\mathcal{J}'} |\alpha_j| = 1 \text{ and } \boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{J}'|}. \tag{4.40}$$

For transitive discretizations the labels need not be restricted to neighborhood $\mathcal{J}_i$ and even more possibilities arise where $\mathcal{J}_i$ consists of values in a ring structure pointing to all other nodes in the domain.

$$\mathcal{J}_i := \Delta_i^{-1}\mathcal{I}. \tag{4.41}$$

Because $|\mathcal{I}|$ will be a substantial number, storing α densely is not a practical solution. Instead, a sparse stencil notation analogous to the one presented for differential operators will be introduced in the next section.

After the uniform discretization is introduced, which will turn out to be both anti-symmetric and transitive, it will become apparent that it has the great advantage of a reduced complexity for these transformations.

The operator overview can now be extended to include finite difference operators. Introducing sub atoms and some ambiguity in the conventions. Depending on the discretization certain differential atoms can only be approximated by finite difference molecules. And the differential atom *id* can be approximated by the finite difference sub atom σ^0:

	Differential operators	Finite Difference operators
Solution	u	$\mathbf{u}$
Subatom	n.a.	ς^j
(Sub)atom	id	$\varsigma^0 = \text{id}$
Atom	$\mathcal{D}_{\mathbf{r}_{i,j}}$	σ^j
Atom/Molecule	$\frac{\partial}{\partial q_k}$	$\sigma_{(\cdot)q_k}$
Atom/Molecule	$\mathcal{D}_{\mathbf{v}}$	$\sigma_{(\cdot)\mathbf{v}}$
Molecule	L	S

4.3. Stencil representation.

A finite difference stencil can be viewed as a sparse notation for the matrices that arise in the finite difference method.

Analogously to section 4 a finite difference atom, ς_j, can be defined using a tuple of scalar $s = 1$ and coefficient in the form of an adjacency label j.

$$\varsigma^j := \{(1, j)\}. \tag{4.42}$$

Scaling the subatomic part with a scalar s yields:

$$s\varsigma^j := \{(s, j)\}. \tag{4.43}$$

A subatomic part scaled by a function $s : \mathcal{I} \to \mathbb{R}$ is denoted as $\{(s(\cdot), j)\}$ where the domain of s for finite difference operators is changed from Ω to the node index set $\mathcal{I}$ such that $s : \mathcal{I} \to \mathbb{R}$. $(\cdot)$ will be omitted if the function character of s is apparent from context.

Applying a single scalar tuple to a discretized function $\mathbf{f}$ yields:

$$(\{(s, j)\}\mathbf{f})_i = \left(s\varsigma^j\right)_i = s\mathbf{f}_{i+\Delta_i j} \tag{4.44}$$

and for a function scalar $s(\cdot)$:

$$(\{(s(\cdot), j)\}\mathbf{f})_i = \left(s(i)\varsigma^j\mathbf{f}\right)_i = s(i)\mathbf{f}_{i+\Delta_i j}. \tag{4.45}$$

Note that using the notation for discretization of function $s, \mathbf{s}^{\mathcal{I}} =: \mathbf{s}$, this can also be denoted as:

$$(\{(s(\cdot), j)\}\mathbf{f})_i = \left(\operatorname{diag}(\mathbf{s})\varsigma^j\right)_i = \mathbf{s}_i\mathbf{f}_{i+\Delta_i j}. \tag{4.46}$$

The setwise union of tuples of the stencil representation is equivalent to the summation of the operators such that $s_1\varsigma^{j_1} + s_2\varsigma^{j_2}$ can be represented by:

$$s_1\varsigma^{j_1} + s_2\varsigma^{j_2} = \{(s_1, j_1), (s_2, j_2)\}. \tag{4.47}$$

General finite difference molecule operators are defined as:

$$S := \bigcup_k \{(s_k, j_k)\}. \tag{4.48}$$

applying them gives:

$$(S\mathbf{f})_i := \sum_k s_k\varsigma^{j_k}. \tag{4.49}$$

The finite difference atom σ^j can now be represented by:

$$\sigma^j = \{(\frac{1}{h_j(\cdot)}, j), (\frac{-1}{h_j(\cdot)}, 0)\}, \tag{4.50}$$

where the function h_j is indeed such that $h_j : \mathcal{I} \to \mathbb{R}$.

The application of a finite difference atom yields:

$$\begin{aligned}
\left(\sigma^j\mathbf{f}\right)_i &= \left(\{(\frac{1}{h_j}, j), (\frac{-1}{h_j}, 0)\}\mathbf{f}\right)_i && (4.51)\\
&= \frac{1}{h_j(i)}\mathbf{f}_{i+\Delta_i j} - \frac{1}{h_j(i)}\mathbf{f}_{i+\Delta_i 0} && (4.52)\\
&= \frac{\mathbf{f}_{i+\Delta_i j} - \mathbf{f}_i}{h_j(i)}, && (4.53)
\end{aligned}$$

which conforms to its implicit definition in (4.31).

The stencil operations of scaling, addition, subtraction and composition are defined in Appendix1 with an adaption to the definition for the composition as it was defined for differential operators. For general finite difference operators composition is defined as:

$$\left(\bigcup_k \{(s_k, j_k)\}\right) \circ \left(\bigcup_{k'} \{(s'_{k'}, j'_{k'})\}\right) := \bigcup_k \bigcup_{k'} \{(s_k s'_{k'}, j_k \Delta_{i+\Delta_i j_k} j'_{k'})\}. \tag{4.54}$$

The new scalars will be formed by the multiplication of all combinations of the scalars. The new coefficients are best viewed as the adjacency label relative from node i of the node that is the j' neighbor of the j neighbor of i.

Only for *transitive* domain discretizations this reduces to the form defined for the differential operators

$$\left(\bigcup_k \{(s_k, j_k)\}\right) \circ \left(\bigcup_{k'} \{(s'_{k'}, j'_{k'})\}\right) := \bigcup_k \bigcup_{k'} \{(s_k s'_{k'}, j_k + j'_{k'})\}. \tag{4.55}$$

Advocating once more to the benefits of the, amongst other properties, transitive uniform discretization.

4.4. Local truncation error.

The local truncation error is the difference between the approximation and the exact derivation of the differential operator. It is, at first, defined per node:

DEFINITION 10. *The* ***local truncation error*** *is defined as*

$$\tau_{u,i} = Lu(\mathbf{q}_i) - (S\mathbf{u})_i \qquad \forall i \in \mathcal{I}. \tag{4.56}$$

It depends on the value of the function in that node and the surrounding nodes:

$$= \quad Lu(\mathbf{q}_i) - \sum_{(s,j)\in S} s\mathbf{u}_{i+\Delta j}. \tag{4.57}$$

Assuming that the approximation of u is exact at $\mathbf{q}$ and using the Taylor series differential operator $T_{\mathbf{a}}(\mathbf{q})$ to estimate the function values of the surrounding nodes, a definition depending solely on the function in node i can be made:

$$= \quad Lu(\mathbf{q}_i) - \sum_{(s,j)\in S} sT_{\mathbf{r}_{i,j}} u(\mathbf{q}_i) \tag{4.58}$$

with

$$T_{\mathbf{a}} := \sum_{n_1=0}^{\infty} \cdots \sum_{n_d=0}^{\infty} \frac{(a_1)^{n_1} \dots (a_d)^{n_d}}{n_1! \dots n_d!} \left(\frac{\partial^{n_1+\dots+n_d}}{\partial q_1^{n_1} \dots \partial q_d^{n_d}} \right). \tag{4.59}$$

For a lattice discretization, it follows that $\mathbf{r}_{i,j}$ is independent of i. This provides a τ_u independent of i:

$$\tau_u \quad = \quad Lu - \sum_{(s,j)\in S} sT_{\mathbf{r}_j} u, \tag{4.60}$$

which can be estimated in terms of a maximum over $\mathcal{Q}$.

From this definition, the notion arises that a smaller distance between nodes provides a smaller truncation error and consequently a better approximation.

The lattice property even allows defining τ as the difference between the operators independent of u, which can be estimated by the order of the $\mathbf{r}_j$ terms used:

Definition 11. *The **local truncation error estimate** is given by the order estimation of the difference between a differential operator and the Taylor sequence approximation of the finite difference operator:*

$$\tau = L - \sum_{(s,j)\in S} sT_{\mathbf{r}_j} = \sum_{k=1}^{d_{\mathbf{q}}} \sum_{j\in\mathcal{J}} \mathcal{O}\left(\mathbf{r}_j\right)_k . \tag{4.61}$$

Note that for the identity label, the Taylor operator is indeed the identity operator:

$$T_{\mathbf{r}_0} = T_{\mathbf{0}} = \sum_{n_1=0}^{\infty} \dots \sum_{n_d=0}^{\infty} \frac{0^{n_1}\dots 0^{n_d}}{n_1!\dots n_d!}\left(\frac{\partial^{n_1+\dots+n_d}}{\partial q_1^{n_1}\dots \partial q_d^{n_d}}\right) \tag{4.62}$$

$$= \frac{0^0\dots 0^0}{0!\dots 0!}\left(\frac{\partial^{0+\dots+0}}{\partial q_1^0\dots\partial q_d^0}\right) \tag{4.63}$$

$$= \frac{1\dots 1}{1\dots 1}\mathrm{id} = \mathrm{id}. \tag{4.64}$$

4.5. Approximating Higher order derivatives. Higher order derivatives can be approximated by composing multiple finite difference operators:

$$\bigcirc_{i=1}^{n} \sigma_{(\cdot)q_{k_i}} \approx \bigcirc_{i=1}^{n} \frac{\partial}{\partial q_{k_i}}. \tag{4.65}$$

The one dimensional discretization of the second order derivative $\frac{\partial^2}{\partial x^2}$ can therefore be approximated by for example using the forward operator twice:

$$\left(\left(\sigma_{Fx}\circ\sigma_{Fx}\right)\mathbf{u}\right)_i = \left(\sigma_{Fx}\left(\frac{\mathbf{u}_{i+\Delta_i 1}-\mathbf{u}_i}{h_1(i)}\right)\right)_i$$

$$= \frac{1}{h_1(i)}\left(\sigma_{Fx}\left(\mathbf{u}_{i+\Delta_i 1}-\mathbf{u}_i\right)\right)_i \tag{4.66}$$

$$= \frac{1}{h_1(i)}\left(\frac{\mathbf{u}_{i+\Delta_i 1+\Delta_{i+\Delta_i 1}1}-\mathbf{u}_{i+\Delta_i 1}}{h_1(i)} - \frac{\mathbf{u}_{i+\Delta_i 1}-\mathbf{u}_i}{h_1(i)}\right)_i . \tag{4.67}$$

For the transitive discretization used in lemma 1 this reduces to

$$= \frac{1}{h_1(i)}\left(\frac{\mathbf{u}_{i+2}-\mathbf{u}_{i+1}}{h_1(i)} - \frac{\mathbf{u}_{i+1}-\mathbf{u}_i}{h_1(i)}\right)_i , \tag{4.68}$$

which can be written as:

$$= \frac{1}{h_1(i)}\left(\left(\frac{\varsigma^2-\varsigma^1}{h_1(i)} - \frac{\varsigma^1-\varsigma^0}{h_1(i)}\right)\mathbf{u}\right)_i \tag{4.69}$$

$$= \left(\frac{\varsigma^2-\varsigma^0}{\left(h_1(i)\right)^2}\mathbf{u}\right)_i . \tag{4.70}$$

Now an explicit approximation of $\frac{\partial^2}{\partial x^2}$ can be written as:

$$\frac{\partial^2}{\partial x^2} \approx \operatorname{diag}(\mathbf{h}_1)^{-2}\left(\varsigma^2 - \varsigma^0\right). \tag{4.71}$$

For a second order d-dimensional derivative over a transitive discretization

$$\operatorname{diag}(\mathbf{h}_j)^{-2}\left(\varsigma^{2j} - \varsigma^0\right) \approx \mathcal{D}_{\mathbf{r}_{i,j}}\mathcal{D}_{\mathbf{r}_{i,j}} \tag{4.72}$$

locally around node i and globally for lattices:

$$\operatorname{diag}(\mathbf{h}_j)^{-2}\left(\varsigma^{2j} - \varsigma^0\right) \approx \mathcal{D}_{\mathbf{r}_j}\mathcal{D}_{\mathbf{r}_j}. \tag{4.73}$$

For a spanning domain discretization, higher order derivatives in all directions can be approximated. Given a vector $\mathbf{v}$ and, following (4.39), the solution $\boldsymbol{\alpha}$ can be written as:

$$\begin{aligned}
\mathcal{D}^2_{\mathbf{v}} &\approx \sigma_{\boldsymbol{\alpha}} \circ \sigma_{\boldsymbol{\alpha}} && (4.74)\\
&= \sum_{j_1\in\mathcal{J}}\sum_{j_2\in\mathcal{J}} \alpha_{j_1}\alpha_{j_2}\sigma^{j_1}\sigma^{j_2} && (4.75)\\
&= \sum_{j_1\in\mathcal{J}}\sum_{j_2\in\mathcal{J}} \alpha_{j_1}\alpha_{j_2}\operatorname{diag}(\mathbf{h}_{j_1})^{-1}\left(\varsigma^{j_1}-\varsigma^0\right)\operatorname{diag}(\mathbf{h}_{j_2})^{-1}\left(\varsigma^{j_2}-\varsigma^0\right). && (4.76)
\end{aligned}$$

For a transitive domain discretization this yields:

$$= \sum_{j_1\in\mathcal{J}}\sum_{j_2\in\mathcal{J}} \alpha_{j_1}\alpha_{j_2}\operatorname{diag}(\mathbf{h}_{j_1})^{-1}\operatorname{diag}(\mathbf{h}_{j_2})^{-1}\left(\varsigma^{j_1+j_2}-\varsigma^{j_1}-\varsigma^{j_2}+\varsigma^0\right). \tag{4.77}$$

A very generalized definition for finite difference approximations of higher order derivatives on transitive spanning domain discretizations is:

$$\bigcirc_{k=1}^{n} \mathcal{D}_{\mathbf{v}_k} \approx \bigcirc_{k=1}^{n} \sigma_{\boldsymbol{\alpha}_k} \tag{4.78}$$

$$= \sum_{j_1\in\mathcal{J}} \cdots \sum_{j_n\in\mathcal{J}} \left(\prod_{k=1}^{n}(\boldsymbol{\alpha}_k)_{j_k}\operatorname{diag}(\mathbf{h}_{j_k})^{-1}\right)\left(\prod_{k=1}^{n}\sigma^{j^k}\right), \tag{4.79}$$

where various choices for the different $\boldsymbol{\alpha}_k$'s yield a very wide scope of approximation options. Truncation error analysis will yield favorable choices that ensure smaller order truncation errors.

4.6. Fractional labels. The options for approximating higher order derivatives are extended even further by the the introduction of fractional labels.

To approximate one dimensional diffusion $\frac{\partial^2}{\partial q_k^2}$ one could use $S^2_{F_x}, S^2_{B_x}$ or $S^2_{C_x}$ but also a mixed approach: $S_{F_x}S_{B_x}$.

On a transitive grid for which it follows that $(i+\Delta_i j)+\Delta_{i+\Delta_i j} j = i+\Delta_i(j+j) = i+\Delta_i(2j)$ and inspired by (4.73) one could use the label $\frac{j}{2}$ yielding:

$$\operatorname{diag}(\mathbf{h}_{\frac{j}{2}})^{-2}\left(\varsigma^{2\frac{j}{2}} - \varsigma^0\right) \approx \mathcal{D}_{\mathbf{r}_{\frac{j}{2}}}\mathcal{D}_{\mathbf{r}_{\frac{j}{2}}}. \tag{4.80}$$

If one intuitively defines the relative position $\mathbf{r}_{\frac{j}{2}}$ to be halfway the relative position of j with respect to the local node i this will yield $\mathbf{r}_{\frac{j}{2}} := \frac{1}{2}\mathbf{r}_j$ and consequently $\mathbf{h}_{\frac{j}{2}} = \frac{1}{2}\mathbf{h}_j$. Noting that scaling $\mathbf{r}_j$ by a positive scalar does not change the directional derivative. (i.e. $\mathcal{D}_{\mathbf{r}_j} = \mathcal{D}_{\frac{1}{2}\mathbf{r}_j}$) and so:

$$(\sigma^{\frac{j}{2}})^2 = \operatorname{diag}(\frac{1}{2}\mathbf{h}_j)^{-2}\left(\varsigma^j - \varsigma^0\right) \approx \mathcal{D}^2_{\mathbf{r}_j}. \tag{4.81}$$

$(\sigma^{\frac{j}{2}})^2$ is a good candidate at being a better approximation of $\frac{\partial^2}{\partial x^2}$ than $(\sigma^j)^2$ because of a reduced truncation error.

The fractional notation $\sigma^{\frac{j}{n}}$ is a solution for the single label σ^j operator. The combined label notation σ_α will run into difficulties when denoted as $\sigma_{\frac{\alpha}{n}}$. It will therefore be denoted as $\sigma_{\alpha^{\frac{1}{n}}}$.

4.7. Operator discretization algorithms. Because of the wide array of approximation options arising in higher order higher dimensional finite difference operators one should carefully denote the methods used and combined.

A common notation for these choices are the BT,FT,CS,FS,BS definitions, where FT and BT indicate the forward, respectively backward, approximation of the time derivative $\frac{\partial}{\partial t} = \frac{\partial}{\partial q_{d_\mathbf{q}}}$ and analogously the CS,FS,BS are the central, forward and backward approximations of the space derivatives: $\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_{d_\mathbf{x}}} = \frac{\partial}{\partial q_1}, \ldots, \frac{\partial}{\partial q_{d_\mathbf{x}}}$.

Unless otherwise stated the second order derivatives will be approximated using the squared semi central operator, $\left(\sigma_{(Cx_k)^{\frac{1}{2}}}\right)^2 \approx \frac{\partial^2}{\partial x^2}$.

Example 6. *BTCS discretization of the test cases*

A BTCS discretization of the test case equations (cf. Appendix 3) yields:

Testcase	*Differential Molecule (L)*		*Finite Difference Molecule(S)*
1	$d\frac{\partial}{\partial t} - \Delta$	$\rightsquigarrow$	$d\sigma_{Bt} - \sum_{k=1}^{d_\mathbf{x}}(\sigma_{(Cx_k)^{\frac{1}{2}}})^2$
2	$\frac{\partial}{\partial t} - \mathbf{x}\cdot\nabla$	$\rightsquigarrow$	$\sigma_{Bt} - \sum_{k=1}^{d_\mathbf{x}} \operatorname{diag}(\mathbf{x}_k)\sigma_{Cx_k}$
3	$d\frac{\partial}{\partial t} + u\,\mathbf{1}\cdot\nabla - \mu\Delta$	$\rightsquigarrow$	$d\sigma_{Bt} + \sum_{k=1}^{d_\mathbf{x}}\left(\operatorname{diag}(\mathbf{u})\sigma_{Cx_k} - \mu(\sigma_{(Cx_k)^{\frac{1}{2}}})^2\right)$

5. Linear System

5.1. Matrix representation. This section focusses on the conversion from a finite difference stencil, S, to a matrix M and a constant term vector $\mathbf{b}$ that define a linear system (LS) from which an approximation to the discretized solution function $\mathbf{u}$ can be obtained.

$$(L, f) \to (S, \mathbf{f}) \to (M, \mathbf{b}). \tag{5.1}$$

As with the conversion from linear differential operators to finite difference operators the approach will be based on handling the atomic parts of the operators first and transforming those:

$$\frac{\partial}{\partial q_k} \rightsquigarrow \sigma_{(\cdot)q_k} \quad \rightsquigarrow \quad D_{(\cdot)q_k}. \tag{5.2}$$

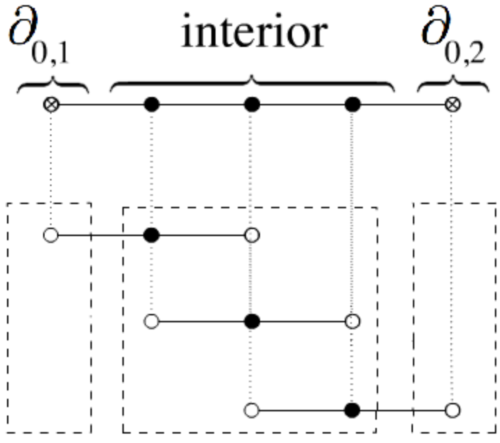
It would be tempting to begin with defining the explicit matrix representation of a sub atomic operator as

$$\varsigma^j = \{(1, j)\} = Q^j \tag{5.3}$$

For some matrix Q^j such that

$$\varsigma^j \mathbf{u} = \{(1, j)\}\mathbf{u} = Q^j \mathbf{u}. \tag{5.4}$$

Matrix Q^j would describe the shift that is induced by retrieving the j-th neighbor for each node i but there are complications. The boundary value problem defines a differential operator L for each boundary subdomain. Suppose that for a one dimensional domain L is a first order derivative that is applied to the interior of the domain. But if $L = \frac{\partial}{\partial x}$ is approximated using a central finite difference operator applying this operator to the interior requires contributions from the boundaries as well:



The application of the central operator to the interior of the domain is described not by one but by three matrices. Their dimensions are indicated by the size of the target domain (the interior) and the source domains (interior and boundaries). Given a target domain $\mathcal{A}$, a source domain $\mathcal{B}$, and an adjacency label j the matrix Q is defined as:

$$Q^j_{\mathcal{A},\mathcal{B}} := \underset{i\in\mathcal{A}}{\boxed{\rightarrow}}\underset{g\in\mathcal{B}}{\boxed{\downarrow}}\ \delta_{i,g+\Delta_i j} \tag{5.5}$$

such that the actual matrix representation of ς^j is given by:

$$\varsigma^j \mathbf{u}^{\mathcal{A}} = \sum_{k,l} Q^j_{\mathcal{A},\partial_{k,l}\mathcal{A}} \mathbf{u}^{\partial_{k,l}\mathcal{A}}. \tag{5.6}$$

Q^j

The matrix representation Q^j of the sub atomic finite difference operator ς^j is a matrix such that at row i it is one exactly at the column position of the j-th neighbor of i and zero elsewhere.

Note that, as the superscript notation of j might suggest:

$$Q^0_{\mathcal{A},\mathcal{B}} := \underset{i\in\mathcal{A}}{\boxed{\rightarrow}}\underset{g\in\mathcal{B}}{\boxed{\downarrow}}\ \delta_{i,g+\Delta_i 0} = \underset{i\in\mathcal{A}}{\boxed{\rightarrow}}\underset{g\in\mathcal{B}}{\boxed{\downarrow}}\ \delta_{i,g}, \tag{5.7}$$

which is equal to the identity matrix, I, if and only if $\mathcal{A} = \mathcal{B}$ and equal to the zero matrix, $\mathbf{0}$, if $\mathcal{A} \bigcap \mathcal{B} = \emptyset$.

Now a differential operator L, approximated by finite difference operator S can be approximated by matrices M:

$$M^S_{\mathcal{A},\mathcal{B}} = \sum_{(s,j)\in S} sQ^j_{\mathcal{A},\mathcal{B}} \tag{5.8}$$

such that

$$S\mathbf{u}^{\mathcal{A}} = \sum_{k,l} M^S_{\partial_{k,l}\mathcal{A},\mathcal{A}} \mathbf{u}^{\partial_{k,l}\mathcal{A}} \approx Lu|_{\mathcal{A}}. \tag{5.9}$$

Now the linear system (LS) that arises from a finite difference equation (FDE) that approximates a partial difference equation (PDE) can be formulated. Given a PDE

$$Lu = f \tag{5.10}$$

for all $\mathbf{q}$ in subdomain ω and an approximating FDE over the subdomain nodes $\mathcal{A}$:

$$S\mathbf{u}^{\mathcal{A}} = \mathbf{f}^{\mathcal{A}} \tag{5.11}$$

the following linear system (LS) is obtained:

$$\sum_{k,l} M^S_{\mathcal{A},\partial_{k,l}\mathcal{A}} \mathbf{u}^{\partial_{k,l}\mathcal{A}} = \mathbf{f}^{\mathcal{A}}. \tag{5.12}$$

Atomic operators $D^j, D_{(\cdot)q_k}$ and $D_{(\cdot)\mathbf{v}}$ are easily defined as matrices $M^{\sigma^j}, M^{\sigma_{(\cdot)q_k}}$ and $M^{\sigma_{(\cdot)\mathbf{q}}}$.

The operator overview can now be extended to include matrix reprentations:

	Differential operators	FD operators	Matrix operators
Solution	u	$\mathbf{u}$	$\mathbf{u}$
Subatom	n.a.	ς^j	$Q^{\mathbf{j}}$
(Sub)atom	id	id	I
Atom	$\mathcal{D}_{\mathbf{r}_{i,j}}$	σ^j	D^j
Atom/Molecule	$\frac{\partial}{\partial q_k}$	$\sigma_{(\cdot)q_k}$	$D_{(\cdot)q_k}$
Atom/Molecule	$\mathcal{D}_{\mathbf{v}}$	$\sigma_{(\cdot)\mathbf{v}}$	$D_{(\cdot)\mathbf{v}}$
Molecule	L	S	M

5.2. Traversal methods. The linear system (5.12) is in most cases underdetermined: it has $|\partial_{k,l}\mathcal{A}|$ equations and $\sum_{k,l}|\partial_{k,l}\mathcal{A}|$ unknowns. The solution of this problem lies in the first assumption that here exist a subdomain $\mathcal{T}_0$ with finite difference operator S^0 for which (5.12) is not underdetermined: $M^{S^0}_{\mathcal{T}_0,\partial_{k,l}\mathcal{T}_0}$ is nonzero if and only if $\partial_{k,l} = \text{id}$ and zero otherwise such that:

$$S^0\mathbf{u}^{\mathcal{T}_0} = M^{S^0}_{\mathcal{T}_0,\mathcal{T}_0}\mathbf{u}^{\mathcal{T}_0}. \tag{5.13}$$

Candidates for such a subdomain are the zero dimensional boundaries for which no differential can be defined. Such a boundary consists of a single vertex and yields a single node equation for which no shifts are used and no boundaries are implicated. Another candidate is the initial condition which in general is also explicitly defined.

The second assumption is that the subdomains and their finite difference operators can be ordered $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$ according to their dependence on other subdomains: $M^{S}_{\mathcal{T}_n,\mathcal{T}_{n'}}$ is zero for $n < n'$ such that:

$$S^n\mathbf{u}^{\mathcal{T}_n} = \sum_{n'=0}^{n} M^{S^n}_{\mathcal{T}_n,\mathcal{T}_{n'}}\mathbf{u}^{\mathcal{T}_{n'}}. \tag{5.14}$$

Definition 12. *A **traversal** $\{\mathcal{T}_0, \ldots, \mathcal{T}_N\}$ is a partition of the node set $\mathcal{I}$.*

The idea is that the traversal levels $\mathcal{T}_n$ are ordered according to their use of known values. At time step n only values already computed at steps $0, \ldots, n-1$ are used to solve for the unknown values of indices in $\mathcal{T}_n$. This approach is iterated till $n = N$ at which point all values have been solved for.

To check into which traversal level a finite difference operator reaches the **level** function is defined:

Definition 13.

$$\text{level}(i,j) := \{n \mid i + \Delta_i j \in \mathcal{T}_n\} \qquad \forall 0 \le n \le N. \tag{5.15}$$

Definition 14. *A traversal is **valid** for finite difference operators $S^0, \ldots, S^N$ if it holds that for $0 \le n \le N, i \in \mathcal{T}_n$ and $(s,j) \in S^n$ $\text{level}(i,j) = n_j \le n$ with n_j not depending on i.*

For valid traversals the level will coincide for all indices in a given traversal level. There is no need to specify an individual index thus we can denote this by $\text{level}(\mathcal{T}_n, j)$.

Solving the system will become trivial when the current level of unknowns depends solely on previous traversal levels and not on each other, this property is defined as *explicit.*

DEFINITION 15. *A traversal is* ***explicit*** *at level n for finite difference operators S^n if it holds that*

$$level(i, j) = n \Leftrightarrow j = 0 \tag{5.16}$$

for all $i \in \mathcal{I}$ and $(s, j) \in S^n$, and ***implicit*** *otherwise.*

The discretization of (solution) functions over a traversal level is denoted as:

$$\mathbf{u}^n := \mathbf{u}^{\mathcal{I}_n} = \underset{i\in\mathcal{I}_n}{\boxed{\downarrow}}\ u(\mathbf{q}_i). \tag{5.17}$$

5.3. Traversal Extensions. By further subdividing the traversal levels more but smaller systems will provide for more efficient solving. This should be done while retaining the validity of the traversal. In Part 3 a time level traversal is formulated for the uniform domain discretization. Time naturally lends itself to such an approach as the *future* boundary is usually undefined while spatial boundaries are commonly defined in all directions.

Although not explored in this thesis the combined solving of multiple traversal levels with different operators will provide options for solving Neumann boundaries, vector valued solutions and adaptive p-refinement which will be described in Part 4.

Subdividing the traversal levels can also be a way to divide the workload amongst multiple processing cores. The matrix formulation for contributions between traversal levels can function as communication matrices for parallelization.

5.4. Traversing the Linear System. Given a valid traversal $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_N$ with finite difference operators $S^0, S^1, \ldots, S^N$ equation (5.14) combined with the notation introduced in (5.17) is used to solve the FDE for each traversal level n:

$$S^n \mathbf{u}^n = \mathbf{f}^n \tag{5.18}$$

$$\Leftrightarrow \quad \sum_{n'=0}^{n} M^{S^{n'}}_{\mathcal{T}_n,\mathcal{T}_{n'}} \mathbf{u}^{n'} = \mathbf{f}^n. \tag{5.19}$$

Sorting unknowns to the left and knowns to the right results in:

$$\Leftrightarrow \quad M^{S^n}_{\mathcal{T}_n,\mathcal{T}_n} \mathbf{u}^n = -\sum_{n'=0}^{n-1} M^{S^n}_{\mathcal{T}_n,\mathcal{T}_{n'}} \mathbf{u}^{n'} + \mathbf{f}^n. \tag{5.20}$$

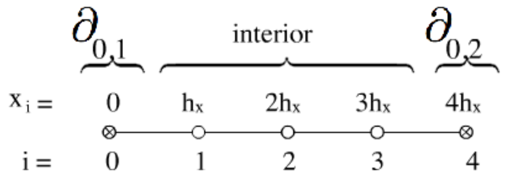
The right hand side can be stored in a right hand side vector $\mathbf{b}^n$:

$$\Leftrightarrow \quad M^{S^n}_{\mathcal{T}_n,\mathcal{T}_n} \mathbf{u}^{\mathcal{T}_n} = \mathbf{b}^n. \tag{5.21}$$

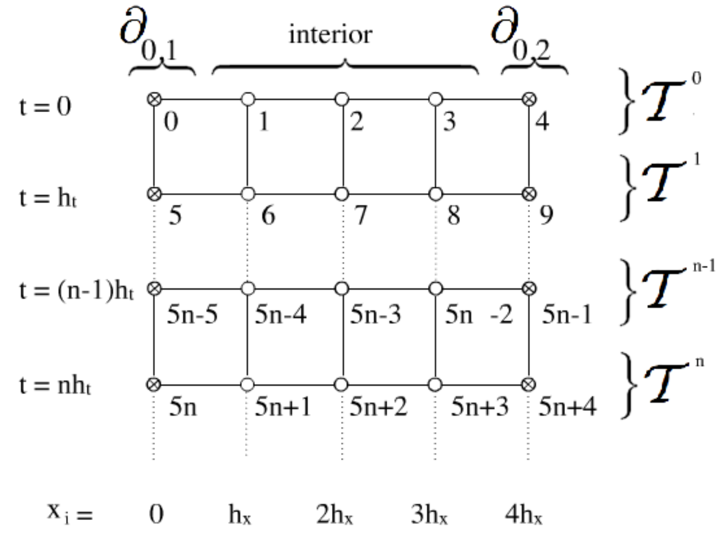
The matrices are constructed by combining the subatomic matrices Q. The construction of these matrices remains implicit. The element values are not computed individually but are constructed by operations on the stencil tuples.

The atomic operators $\sigma^j, \sigma_{(\cdot)\mathbf{q}_k}, D^j, M^j, D_{(\cdot)\mathbf{q}_k}, M_{(\cdot)\mathbf{q}_k}$ are not constructed explicitly instead they are broken apart into their subatomic parts which can be divided amongst the left hand matrix A^n_S and the right hand vector $\mathbf{b}^n$.

Example 7. *BTCS discretization of Test Case I*
Consider a one dimensional domain discretized into five nodes:



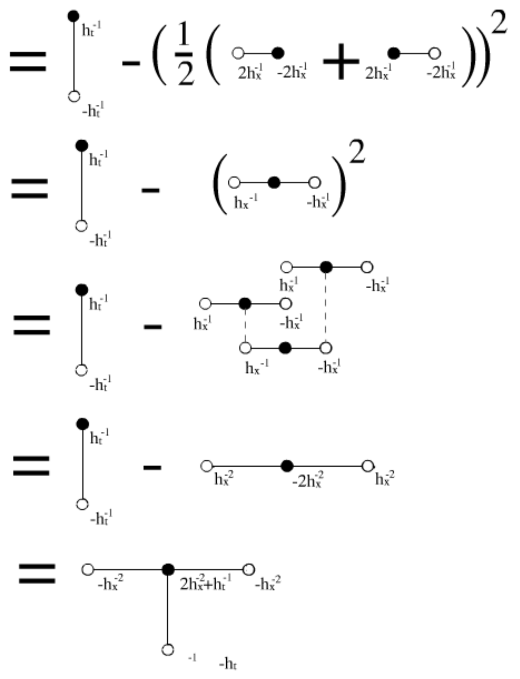
A uniform time traversal yields the following structure:



BTCS Discretization of the Heat equation differential operator yields:

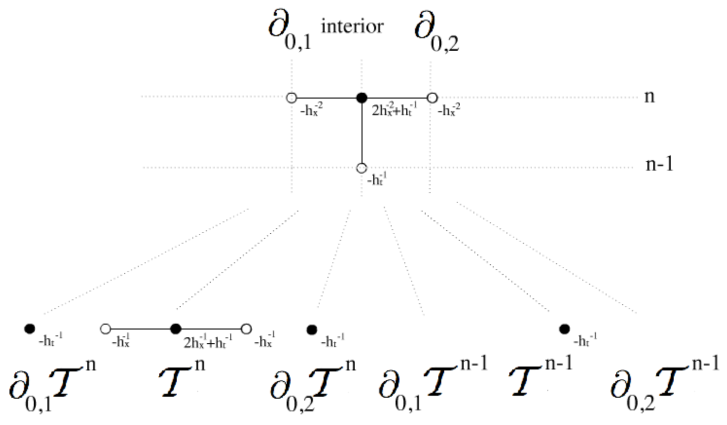
$$\begin{aligned} L &= \frac{\partial u}{\partial t} - \frac{\partial^2}{\partial x^2} \\ &\approx \sigma_{Bt} - \sigma^2_{(Cx)^{\frac{1}{2}}}. \end{aligned}$$

Continuing with a visual representation:

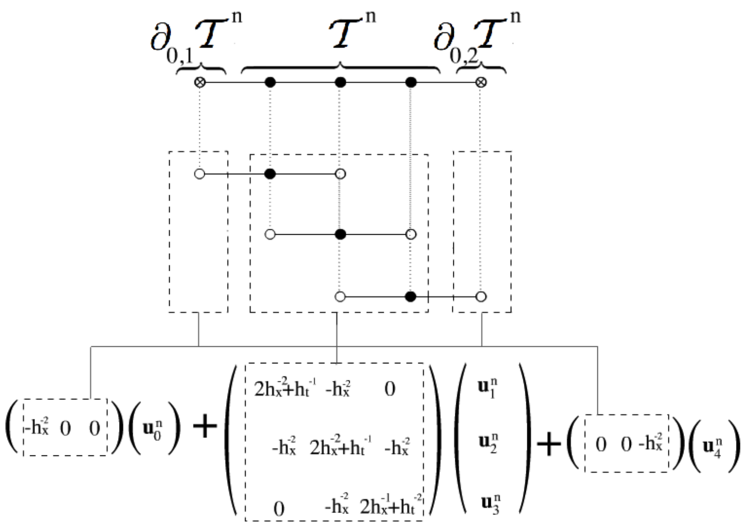


.

Divide the 'influences' of the operator over traversal levels:

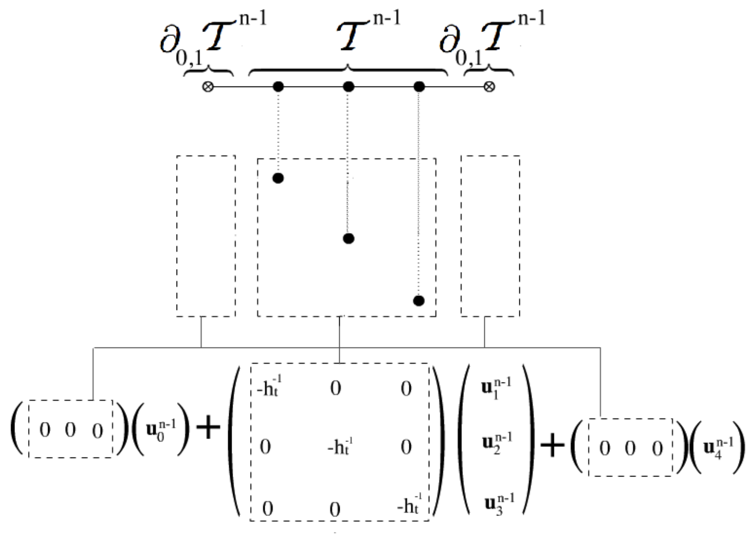


Apply the operator to the n layer:



$$\begin{pmatrix} -h_x^{-2} & 0 & 0 \end{pmatrix}\begin{pmatrix} \mathbf{u}_0^n \end{pmatrix} + \begin{pmatrix} 2h_x^{-2}+h_t^{-1} & -h_x^{-2} & 0 \\ -h_x^{-2} & 2h_x^{-2}+h_t^{-1} & -h_x^{-2} \\ 0 & -h_x^{-2} & 2h_x^{-1}+h_t^{-2} \end{pmatrix}\begin{pmatrix} \mathbf{u}_1^n \\ \mathbf{u}_2^n \\ \mathbf{u}_3^n \end{pmatrix} + \begin{pmatrix} 0 & 0 & -h_x^{-2} \end{pmatrix}\begin{pmatrix} \mathbf{u}_4^n \end{pmatrix}$$

Apply the operator to the $n-1$ layer:



$\partial_{0,1}\mathcal{T}^{n-1}$
$\mathcal{T}^{n-1}$
$\partial_{0,1}\mathcal{T}^{n-1}$
$\begin{pmatrix} 0 & 0 & 0 \end{pmatrix}\begin{pmatrix}\mathbf{u}_0^{n-1}\end{pmatrix} + \begin{pmatrix} -h_t^{-1} & 0 & 0 \\ 0 & -h_t^{-1} & 0 \\ 0 & 0 & -h_t^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{u}_1^{n-1}\\ \mathbf{u}_2^{n-1}\\ \mathbf{u}_3^{n-1}\end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}\begin{pmatrix}\mathbf{u}_4^{n-1}\end{pmatrix}$

Which yields a, perhaps familiar, linear system for the Heat equation:

$$\begin{pmatrix} 2h_x^{-2}+h_t^{-1} & -h_x^2 & 0 \\ -h_x^2 & 2h_x^{-2}+h_t^{-1} & -h_x^2 \\ 0 & -h_x^2 & 2h_x^{-2}+h_t^{-1} \end{pmatrix}\begin{pmatrix} u_1^n \\ u_2^n \\ u_3^n \end{pmatrix}$$

$$= \; -\begin{pmatrix} -h_t^{-1} & 0 & 0 \\ 0 & -h_t^{-1} & 0 \\ 0 & 0 & -h_t^{-1} \end{pmatrix}\begin{pmatrix} u_1^{n-1} \\ u_2^{n-1} \\ u_3^{n-1} \end{pmatrix} - \begin{pmatrix} -h_x^{-2} \\ 0 \\ 0 \end{pmatrix}(u_0^n) - \begin{pmatrix} 0 \\ 0 \\ -h_x^{-2} \end{pmatrix}(u_5^n.)$$

The implications of the *explicit* property of a stencil become apparent when stating the following lemma:

LEMMA 2. *S^n is explicit if and only if $M^{S^n}_{\mathcal{T}_n,\mathcal{T}_n} = \mathrm{diag}(\mathbf{a})I$ with $\mathbf{a} \in \mathbb{R}^{|\mathcal{T}_n|}$.*

Proof: *5.8 states that:*

$$M^{S^n}_{\mathcal{T}_n,\mathcal{T}_n} := \sum_{(s,j)\in S^n} sQ^j_{\mathcal{T}_n,\mathcal{T}_n}. \tag{5.22}$$

Using definition 15:

$$= \sum_{(s,0)\in S^n} sQ^0_{\mathcal{T}_n,\mathcal{T}_n} \tag{5.23}$$

and with the conclusion after :

$$= \sum_{(s,0)\in S^n} sI. \tag{5.24}$$

If all s are scalars, as opposed to a function scalar $s(\cdot)$, define $\mathbf{a} = s\mathbf{1}$ or else $\mathbf{a} := \sum_{\substack{(s,0)\in S^n \\ \mathrm{level}(\mathcal{T}_n,0)=n}} s$ and conclude:

$$= \mathrm{diag}(\mathbf{a})I. \tag{5.25}$$

Solving becomes trivial because the inverse of diagonal matrix $\mathrm{diag}(\mathbf{a})I$ is a diagonal matrix with the inverted elements of $\mathbf{a}$.

5.5. Stability Analysis. The following section focusses on stability analysis. Central in this analysis is the question: what are the consequences of the approximation defects in a given traversal layer? In other words, how do errors propagate?

5.5.1. *Two level.* Consider a two level stencil S, so that for the values of $\mathcal{T}_n$ only values of $\mathcal{T}_{n-1}$ are used and $|\mathcal{T}_n| = |\mathcal{T}_{n-1}|$. Then (5.20) reduces to:

$$\Leftrightarrow \quad M^{S^n}_{\mathcal{T}_n,\mathcal{T}_n}\mathbf{u}^n = -M^{S^n}_{\mathcal{T}_n,\mathcal{T}_{n-1}}\mathbf{u}^{n-1} + \mathbf{f}^n. \tag{5.26}$$

Rename the matrices:

$$A^n\mathbf{u}^n = B^n\mathbf{u}^{n-1} + \mathbf{f}^n, \qquad \forall n \geq 1 \tag{5.27}$$

with A^n a $|\mathcal{T}_n| \times |\mathcal{T}_n|$-matrix and B^n a $|\mathcal{T}_n| \times |\mathcal{T}_{n-1}|$-matrix. The superscript index n might be confused for an exponent but is chosen to conform to the superscript index of $\mathbf{u}^n$. Assuming A^n is non singular, already a prerequisite for a system that can actually be solved, we can state:

$$\mathbf{u}^n = (A^n)^{-1} B^n\mathbf{u}^{n-1} + (A^n)^{-1}\mathbf{f}^n \qquad \forall n \geq 1. \tag{5.28}$$

The system is said to be stable if any perturbation, $\boldsymbol{\epsilon}$, of solution $\mathbf{u}^{n-1}$ is reduced when solving for $\mathbf{u}^n$.

$$(A^n)^{-1} B^n(\mathbf{u}^{n-1} + \boldsymbol{\epsilon}) + (A^n)^{-1}\mathbf{f}^n \tag{5.29}$$

$$= (A^n)^{-1} B^n\mathbf{u}^{n-1} + (A^n)^{-1} B^n\boldsymbol{\epsilon}) + (A^n)^{-1}\mathbf{f}^n. \tag{5.30}$$

Using (5.28) this is equal to:

$$= \mathbf{u}^n + (A^n)^{-1} B^n \boldsymbol{\epsilon}, \qquad \forall n \geq 1. \tag{5.31}$$

For stability, the new perturbation, $(A^n)^{-1} B^n \boldsymbol{\epsilon}$, should be of smaller size than the original perturbation $\boldsymbol{\epsilon}$. That reduction (or amplification in case the constraint is not met) is measured using a norm such that a stability restraint is formulated as:

$$|| (A^n)^{-1} B^n \boldsymbol{\epsilon}|| < ||\boldsymbol{\epsilon}||, \qquad \forall \boldsymbol{\epsilon} \in \mathbb{R}^d. \tag{5.32}$$

This is equivalent to a stability restraint formulated using an induced matrix norm:

$$\left\| (A^n)^{-1} B^n \right\| < 1, \tag{5.33}$$

This restriction is also met by restricting the moduli of the eigenvalues to

$$\left\| \lambda \left((A^n)^{-1} B^n \right) \right\| \subseteq (0, 1). \tag{5.34}$$

The proof of this can be found in Lemma C8. This appendix also contains estimation properties for eigenvalues and matrix norms.

This is again equivalent to a restriction of the eigenvalues to the open complex unit disk, $\mathbb{C}_{<1}$:

$$\lambda \left((A^n)^{-1} B^n \right) \subseteq \mathbb{C}_{<1} := \left\{ a + ib \in \mathbb{C} \mid a^2 + b^2 < 1 \right\}. \tag{5.35}$$

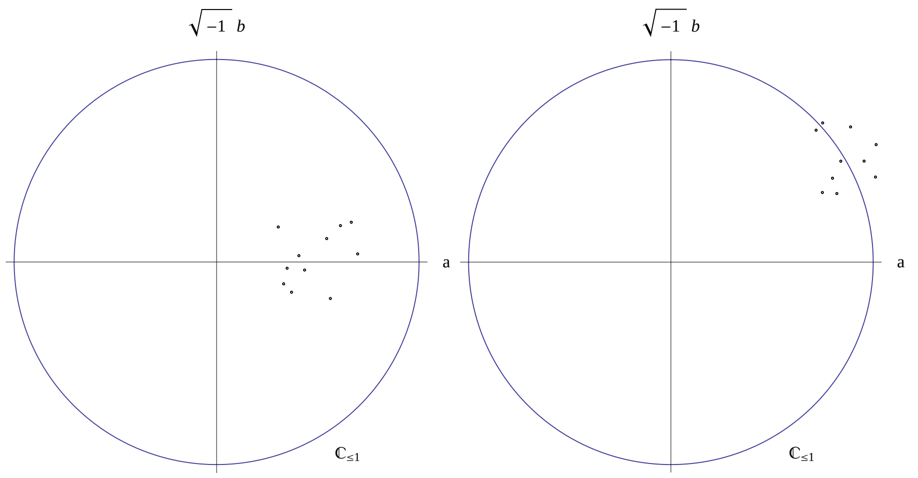


FIGURE 6. (Left) All eigenvalues are within the complex unit disk: the method is stable. (Right) Some eigenvalues are outside the complex unit dis: the method is unstable.

Adhering to the restriction means that any error arising through approximation deficiencies such as the truncation error are not carried forward into next iterations. Alternatively: not adhering to these restriction may cause the iterative process to 'explode':

errors are amplified exponentially and thus very quickly dominate the approximation which thereforebecomes nonsensical.

A discretization is conditionally stable when stability depends on a specific choice of parameters. It is unconditionally (un)stable when the choice of parameters does not influence the restriction.

Calculating the eigenvalues explicitly is in many cases a complex task. Instead, matrix norms are used to estimate an ellipsoidal region containing all eigenvalues and ensuring that this region is restricted to the complex unit disk.

This may result in overestimation of the restrictions and false negatives as shown in Figure 7.

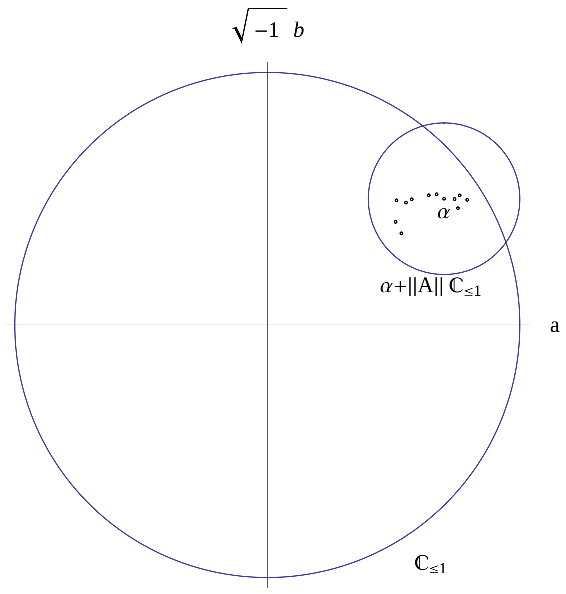


FIGURE 7. All eigenvalues are within the complex unit disk: the method is stable. But the estimation of matrix $\alpha I + A$ using a matrix norm yields a larger region which will result in a false negative for stability check.

Choosing a proper discretization can prove to be helpful to favorably structured matrices for eigenvalue and matrix norm estimates. To some extent the uniform discretization will lend itself to such structures.

5.5.2. *Multi level.* Now consider a k-level stencil S, so that for the values of $\mathcal{T}_n$ only values of $\mathcal{T}_{n-1}, \ldots, \mathcal{T}_{n-k}$ are used and $|\mathcal{T}_n| = |\mathcal{T}_{n-1}| = \ldots = |\mathcal{T}_{n-k}|$. Then (5.20) reduces to:

$$A^n \mathbf{u}^n = \sum_{i=1}^{k} B^n \mathbf{u}^{n-i} + \mathbf{f}^n, \qquad \forall n \geq k \tag{5.36}$$

with B^n a $|\mathcal{T}_n| \times |\mathcal{T}_{n-i}| = |\mathcal{T}_n| \times |\mathcal{T}_n|$-matrix. Assuming A^n is non singular, already a prerequisite for a system that can actually be solved, we can state:

$$\mathbf{u}^n = \sum_{i=1}^{k} (A^n)^{-1} B^n \mathbf{u}^{n-i} + (A^n)^{-1}\mathbf{f}^n \qquad \forall n \geq k \tag{5.37}$$

Introducing perturbations, $\boldsymbol{\epsilon}_i$, in $\mathbf{u}^{n-i}$ results in:

$$\sum_{i=1}^{k} (A^n)^{-1} B^n (\mathbf{u}^{n-i} + \boldsymbol{\epsilon}_i) + (A^n)^{-1}\mathbf{f}^n \tag{5.38}$$

$$= \sum_{i=1}^{k} (A^n)^{-1} B^n \mathbf{u}^{n-i} + (A^n)^{-1}\mathbf{f}^n + \sum_{i=1}^{k} (A^n)^{-1} B^n \boldsymbol{\epsilon}_i \tag{5.39}$$

$$= \mathbf{u}^n + \sum_{i=1}^{k} (A^n)^{-1} B^n \boldsymbol{\epsilon}_i \qquad \forall n \geq k. \tag{5.40}$$

Analogously to the two time level approach, stability is ensured when the perturbations are reduced which is when:

$$\left\| (A^n)^{-1} B^n \right\| < 1, \qquad \forall 1 \leq i \leq k. \tag{5.41}$$

When the $|\mathcal{T}_{n-i}|$'s differ non square matrices will arise. The analysis of these cases is not further explored in this thesis.

5.6. Conclusion for Generalized d-dimensional FDM. After a very generalized approach certain characteristics are shown to potentially facilitate easier (and perhaps more efficient) approximations. For the domain discretization these characteristics include transitivity, anti-symmetry and spanning to facilitate easy operations and proper discretization of the differential operators.All these structures are easy to implement for automatization: given parameters for discretization, a boundary value problem can be converted automatically to a linear system.For the discretization of the differential operators a low truncation error yields higher quality approximations and favorably structured matrices for which eigenvalues can be easily estimated and are stable. These are qualities that can, to an extent, be found in the aforementioned uniform discretization which will be analyzed in the next chapter.

CHAPTER 3

Uniform d-dimensional FDM

1. Introduction

The uniform discretization introduced in this chapter is far from a novel idea. It is extensively used and a very intuitive and practical discretization. By first introducing the concepts for general discretizations I endeavored to formulate formal definitions that are applicable to various domains independent of dimensionality. In this chapter the uniform discretization will be formulated according to those definitions resulting in practical and efficient matrix representation for higher dimensional domains. This will show the formulation to have to following advantages:

- memory and computational efficient application;
- favorable properties to analyse stability;
- stencil multiplication to create higher order derivatives is trivial;
- regularity enhances possibilities for parallelization.

1.1. Uniform Domain Discretization.

Definition 16. *Let $\Omega \subset \mathbb{R}^d$ be defined as the hyper-rectangle $[(q_{\min})_1, (q_{\max})_1] \times \ldots \times [(q_{\min})_d, (q_{\max})_d]$. The **uniform** discretization is defined by distributing $m_1 \cdot \ldots \cdot m_d$ nodes uniformly over this interval while covering the boundary:*

$$\psi(i) := \mathbf{q}_{min} + H\chi(i) \tag{1.1}$$

for $i \in \mathcal{I} = \{1, \ldots, \mathbf{m}^d\}$ where

$$\mathbf{m}^k := m_1 \cdot \ldots \cdot m_k \tag{1.2}$$

$$\mathbf{m}^0 := 1 \tag{1.3}$$

$$\chi(i) := \sum_{k=1}^{d} \left(\lfloor \frac{i-1}{\mathbf{m}^{k-1}} \rfloor \bmod m_k \right) \mathbf{e}_k \tag{1.4}$$

and

$$H := \begin{pmatrix} h_{q_1} & & \\ & \ddots & \\ & & h_{q_d} \end{pmatrix} \tag{1.5}$$

with $h_{q_k} = \frac{(q_{\max}-q_{\min})_k}{m_k-1}$ *nonzero. (If* $h_k = 0$ *then it follows that* $(\mathbf{q}_{\min})_k = (\mathbf{q}_{\max})_k$ *making the hyper rectangular* Ω *de facto of lower-than-d-dimensionality.)*

The adjacency label set and adjacency operator are defined as:

$$\mathcal{J}_i :\equiv \bigcup_{k=1}^{d} \{\pm \mathbf{e}_k\} \tag{1.6}$$

$$\Delta_i(\mathbf{j}) :\equiv \chi^{-1}(\mathbf{j}) = \sum_{k=1}^{d} \mathbf{m}^{k-1} j_k, \qquad \forall i \in \mathrm{int}(\mathcal{I}). \tag{1.7}$$

For a spatially hyper-cubical lattice grids $h_{x_1} = \ldots = h_{x_{d_x}}$ *holds,* h_t *can differ. For most example and test methods a spatially uniform grid will be used, but the method also applies for more general uniform grids.*

LEMMA 3. *The d-dimensional uniform discretization defined in definition 16*

i) is regular and a lattice,

ii) has degree 2d,

iii) is anti-symmetric,

iv) is transitive,

v) and is spanning.

Proof:

i) regularity follows directly from the definition of $\mathcal{J}_i$.

$$\mathbf{r}_{i,\mathbf{j}} = \pm h_k \mathbf{e}_k \tag{1.8}$$

where $k = | < \mathbf{j}, (1, \ldots, d) > |$.

$$h_{\mathbf{j}}(i) = h_k \tag{1.9}$$

$$R_i = \underset{\mathbf{j}\in\mathcal{J}}{\boxed{\rightarrow}} h_{\mathbf{j}}^{-1}(i)\mathbf{r}_{i,\mathbf{j}} \tag{1.10}$$

$$= \overset{d}{\underset{k=1}{\boxed{\rightarrow}}} \begin{pmatrix} h_{\mathbf{e}_k}^{-1}(i)\mathbf{r}_{i,\mathbf{e}_k} & h_{-\mathbf{e}_k}^{-1}(i)\mathbf{r}_{i,-\mathbf{e}_k} \end{pmatrix} \tag{1.11}$$

$$= \overset{d}{\underset{k=1}{\boxed{\rightarrow}}} \begin{pmatrix} \mathbf{e}_k & \mathbf{e}_k \end{pmatrix} \tag{1.12}$$

$$= \overset{d}{\underset{k_1=1}{\boxed{\rightarrow}}} \overset{d}{\underset{k_2=1}{\boxed{\downarrow}}} \begin{pmatrix} (\mathbf{e}_{k_1})_{k_2} & (\mathbf{e}_{k_1})_{k_2} \end{pmatrix} \tag{1.13}$$

$$= \overset{d}{\underset{k_1=1}{\boxed{\rightarrow}}} \overset{d}{\underset{k_2=1}{\boxed{\downarrow}}} (\delta_{k_1,k_2}, \delta_{k_1,k_2}) \tag{1.14}$$

$$= \overset{d}{\underset{k_1=1}{\boxed{\rightarrow}}} \overset{d}{\underset{k_2=1}{\boxed{\downarrow}}} \delta_{k_1,k_2} \begin{pmatrix} 1 & 1 \end{pmatrix} \tag{1.15}$$

$$= I_d \otimes \begin{pmatrix} 1 & 1 \end{pmatrix} \tag{1.16}$$

which is independent of i thus proving the discretization to be a lattice. $\square$

ii) This follows directly from observing that $|\mathcal{J}_i| = 2d$ $\square$

iii)

$$\begin{aligned} \Delta_i\mathbf{j} &= \chi^{-1}(\mathbf{j}) \\ &= \sum_{k=1}^{d} \mathbf{m}^{k-1} j_k \\ &= -\sum_{k=1}^{d} \mathbf{m}^{k-1}(-j_k) \\ &= -\chi^{-1}(-\mathbf{j}) \\ &= -\Delta_i(-\mathbf{j}) \quad \square \end{aligned} \tag{1.17}$$

iv)

$$\begin{aligned} \Delta_i(\mathbf{j}_1 + \mathbf{j}_2) &= \chi^{-1}(\mathbf{j}_1 + \mathbf{j}_2) \\ &= \sum_{k=1}^{d} \mathbf{m}^{k-1}(\mathbf{j}_1 + \mathbf{j}_2)_k \\ &= \sum_{k=1}^{d} \mathbf{m}^{k-1}(\mathbf{j}_1)_k + \sum_{k=1}^{d} \mathbf{m}^{k-1}(\mathbf{j}_2)_k \\ &= \Delta_i\mathbf{j}_1 + \Delta_i\mathbf{j}_2 \\ &= \Delta_i\mathbf{j}_1 + \Delta_{i+\Delta\mathbf{j}_1}\mathbf{j}_2 \quad \square \end{aligned} \tag{1.18}$$

v) The subset of relative positions $\mathbf{r}_{i,1}, \mathbf{r}_{i,3}, \ldots, \mathbf{r}_{i,1+2d_\mathbf{q}}$ *equals* $h_1\mathbf{e}_1, h_2\mathbf{e}_2, \ldots, h_{d_\mathbf{q}}\mathbf{e}_{d_\mathbf{q}}$ *which trivially span* $\mathbb{R}^{d_\mathbf{q}}$ *for* $h_k \neq 0$. □

Transitivity and spanning result in easy and complete discretization of differential operators and easy operations on the resulting finite difference operators to create approximations for higher order derivatives.

1.2. Boundaries. For a point on a k-dimensional boundary of a d-dimensional hyperrectangle, it holds that $d-k$ of its coordinates must be equal to its corresponding value of $\mathbf{q}_{\min}$ or $\mathbf{q}_{\max}$ and thus fixed.

The number of k-dimensional boundaries for a d-dimensional hyper rectangle is given by the number of options for fixing $d-k$ coordinates times the number of options to choose minimal or maximal boundaries.

$$l_k = 2^{d-k}\binom{d}{k}. \tag{1.19}$$

A definition for the l-th k-dimensional boundary of a d-dimensional hyperrectangle is given by:

$$\partial_{k,l}\Omega = \left\{ \mathbf{q} \in \Omega \,\middle|\, \mathbf{q}_{k'} = \begin{cases} (\mathbf{q}_{\min})_{k'} & \left\lfloor \frac{l}{2^{i-1}} \right\rfloor \bmod 2 = 0 \\ (\mathbf{q}_{\max})_{k'} & \left\lfloor \frac{l}{2^{i-1}} \right\rfloor \bmod 2 = 1 \end{cases}, \right. \\ \left. k' = \operatorname{perm}(\left\lfloor \frac{l}{2^{d-k}} \right\rfloor, d-k, d)_i, 1 \leq i \leq d-k \right\}, \tag{1.20}$$

where perm(a, b, c) is the a-th permutation of b numbers from numbers $1, \ldots, c$.

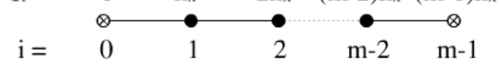
Example 8. *One dimensional Uniform Discretization*

Let $\Omega \subset \mathbb{R}$ *be defined as the interval* $[q_{\min}, q_{\max}]$. *Distributing* m *nodes uniformly over this interval while covering the boundary yields:*

$$q_i := \psi(i) = q_{\min} + h(i-1), \tag{1.21}$$

where $h = \frac{q_{\max}-q_{\min}}{m-1}$.

The adjacency label set is defined independent of i *as* $\mathcal{J}_i \equiv \{-1, 1\}$, *the adjacency operator is defined as* $\Delta_i(j) :\equiv j$ *for all* $i \in \operatorname{int}(\mathcal{I})$.



EXAMPLE 9. *Example: Two dimensional Uniform Discretization*

Let $\Omega \subset \mathbb{R}^2$ be defined as the rectangle $[(q_{\min})_1, (q_{\max})_1] \times [(q_{\min})_2, (q_{\max})_2]$. Distributing $m_1 m_2$ nodes uniformly over this interval while covering the boundary yields:

$$\mathbf{q}_i := \psi(i) = \mathbf{q}_{\min} + H\chi(i), \tag{1.22}$$

where $H = \begin{pmatrix} h_{q_1} & 0 \\ 0 & h_{q_2} \end{pmatrix}$ *with* $h_{q_k} = \frac{(q_{\max} - q_{\min})_k}{m_k - 1}$.

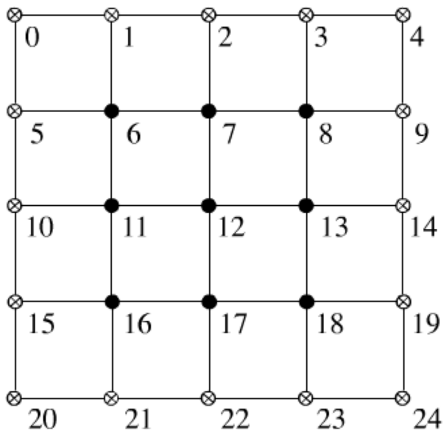
and

$$\chi(i) := \begin{pmatrix} (i-1) \bmod m_1 \\ \lfloor \frac{i-1}{m_1} \rfloor \end{pmatrix} \tag{1.23}$$

The adjacency label set is defined independent of i as

$$\mathcal{J}_i :\equiv \{\begin{pmatrix} \pm 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \pm 1 \end{pmatrix}\} \tag{1.24}$$

for all $i \in \operatorname{int}(\mathcal{I})$ and for those i the adjacency operator is defined as $\Delta_i(\mathbf{j}) := \chi^{-1}(\mathbf{j})$. The discretization defines a square lattice if $h_1 = h_2$.



2. Uniform finite difference operators

Analogously to the definitions for the one dimensional case in 4.2.2 using the anti-symmetric property of the uniform domain discretization for higher dimensional spaces $\frac{\partial}{\partial q_k}$ can be approximated by two different finite difference atoms: a forward and backward finite difference operator (F respectively B) can be defined:

$$\sigma_{Fq_k} := \varsigma^{\mathbf{e}_k} \approx \frac{\partial}{\partial q_k} \tag{2.1}$$

$$\sigma_{Bq_k} := \varsigma^{-\mathbf{e}_k} \approx \frac{\partial}{\partial q_k} \tag{2.2}$$

A weighted average of these two approximations results in the first order θ finite difference operator:

$$\sigma_{\Theta q_k(\theta)} = \theta\varsigma^{\mathbf{e}_k} + (1-\theta)\varsigma^{-\mathbf{e}_k} \quad \approx \quad \frac{\partial}{\partial q_k} \tag{2.3}$$

For $0 \leq \theta \leq 1$ with special case $\theta = \frac{1}{2}$, the central finite difference operator, C, can be defined:

$$\sigma_{Cq_k} \quad := \quad \sigma_{\Theta q_k(\frac{1}{2})} = \frac{\varsigma^{\mathbf{e}_k} + \varsigma^{-\mathbf{e}_k}}{2} \approx \frac{\partial}{\partial q_k} \tag{2.4}$$

Adhering to the $\sigma_{\boldsymbol{\alpha} q_k}$ formulation I derive the following explicit values for $F, B, \Theta(\theta)$ and C:

$$Fq_k \quad := \quad \mathbf{e}_{2k} \tag{2.5}$$
$$Bq_k \quad := \quad \mathbf{e}_{2k+1} \tag{2.6}$$
$$\Theta(\theta)q_k \quad := \quad \theta\mathbf{e}_{2k} + (1-\theta)\mathbf{e}_{2k+1} \tag{2.7}$$
$$Cq_k \quad := \quad \Theta(\frac{1}{2})q_k = \frac{\mathbf{e}_{2k} + \mathbf{e}_{2k+1}}{2}. \tag{2.8}$$

Ordering the labels of $\mathcal{J}$ accordingly results in

$$R \quad = \quad I_d \otimes \begin{pmatrix} 1 & -1 \end{pmatrix}. \tag{2.9}$$

This means that the solution to

$$R\boldsymbol{\alpha} = \frac{\mathbf{v}}{||\mathbf{v}||} \tag{2.10}$$
$$\Leftrightarrow I_d \otimes \begin{pmatrix} 1 & -1 \end{pmatrix} \boldsymbol{\alpha} = \frac{\mathbf{v}}{||\mathbf{v}||} \tag{2.11}$$

can be split $\boldsymbol{\alpha}$ into $\boldsymbol{\alpha}_F$ containing all α_k's for k is odd and $\boldsymbol{\alpha}_B$ containing all α_k's for k is even. They correlate to labels $\mathbf{e}_k$ respectively $-\mathbf{e}_k$:

$$\boldsymbol{I}\boldsymbol{\alpha}_F - \boldsymbol{I}\boldsymbol{\alpha}_B = \frac{\mathbf{v}}{||\mathbf{v}||}. \tag{2.12}$$

To approximate a directional derivative $\mathcal{D}_\mathbf{v}$ by directional finite difference operator $\sigma_{(\cdot)\mathbf{v}}$ we can solve for $\boldsymbol{\alpha}_F$ while setting $\boldsymbol{\alpha}_B = \mathbf{0}$ finding $\boldsymbol{\alpha}_F = \mathbf{v}$ to obtain a forward directional finite difference operator $\sigma_{F\mathbf{v}}$ and vice versa for the backward directional finite difference operator, $\sigma_{B\mathbf{v}}$, with $\boldsymbol{\alpha}_B = \mathbf{v}$. Again with θ and thus central extension possibilities. Using the ordering it follows that: $(F\mathbf{v})_{2k} = -(B\mathbf{v})_{2k+1}$.

The transformation of derivatives turns out to be trivial for the uniform discretization. The transitivity property also ensures trivial operator composition to create approximations to higher order derivatives.

2.1. Uniform local truncation error. The *local truncation error estimate* from definition 11 is easily shown to be even simpler for uniform discretizations:

$$\tau = L - \sum_{(s,j)\in S} sT_{\mathbf{r}_j} = \sum_{k=1}^{d_{\mathbf{q}}} \sum_{j\in\mathcal{J}} \mathcal{O}\left(\mathbf{r}_j\right)_k = \sum_{k=1}^{d} \mathcal{O}\left(h_k\right). \tag{2.13}$$

The truncation error for a uniform label $\mathbf{j} = \pm\mathbf{e}_k$ is greatly simplified compared to the general definition. Because of the reduced complexity of the Taylor series (4.62):

$$T_{\mathbf{r}_{\pm\mathbf{e}_k}}(\mathbf{q}) = T_{\pm h_k \mathbf{e}_k}(\mathbf{q}) \tag{2.14}$$

$$= \sum_{n_1=0}^{\infty} \cdots \sum_{n_d=0}^{\infty} \frac{0^{n_1}\ldots(\pm h_k)^{n_k}\ldots 0^{n_d}}{n_1!\ldots n_d!} \left(\frac{\partial^{n_1+\ldots+n_d}}{\partial q_1^{n_1}\ldots\partial q_d^{n_d}}\right). \tag{2.15}$$

Using

$$0^{n_k} = \begin{cases} 1 & n_k = 1 \\ 0 & \text{otherwise} \end{cases}, \frac{\partial^0}{\partial q_k^0} = \text{id}, \text{ and } 0! = 1 \tag{2.16}$$

(2.14) reduces to:

$$= \sum_{n=0}^{\infty} \frac{(\pm h_k)^n}{n!} \left(\frac{\partial^n}{\partial q_k^n}\right) \tag{2.17}$$

For transitive labels $\mathbf{j}_1 + \mathbf{j}_2$, possibly mixed from approximating mixed derivatives, the Taylor series for the uniform domain discretization will show to still be quite manageable.

The truncation errors for the forward, backward and central finite difference operators are computed below.

Forward finite difference operator σ_{Fq_k}:

$$\tau = \frac{\partial}{\partial q_k} - \sum_{(s,j)\in\sigma_{Fq_k}} sT_{\mathbf{r}_j} \tag{2.18}$$

$$= \frac{\partial}{\partial q_k} - h_{q_k}^{-1} T_{h_{q_k}\mathbf{e}_k} + h_{q_k}^{-1} T_{\mathbf{0}} \tag{2.19}$$

$$= \frac{\partial}{\partial q_k} - h_{q_k}^{-1} \sum_{n=0}^{\infty} \frac{h_{q_k}^n}{n!} \frac{\partial^n}{\partial q_k^n} + h_{q_k}^{-1} \sum_{n=0}^{\infty} \frac{0^n}{n!} \frac{\partial^n}{\partial q_k^n} \tag{2.20}$$

$$= \frac{\partial}{\partial q_k} - h_{q_k}^{-1}\text{id} - \frac{\partial}{\partial q_k} - h_{q_k}^{-1} \sum_{n=2}^{\infty} \frac{h_{q_k}^n}{n!} \frac{\partial^n}{\partial q_k^n} + h_{q_k}^{-1}\text{id} \tag{2.21}$$

$$= -\sum_{n=2}^{\infty} \frac{h_{q_k}^{n-1}}{n!} \frac{\partial^n}{\partial q_k^n} = \mathcal{O}(h_{q_k}). \tag{2.22}$$

This shows the forward finite difference operator to be of first order.

Backward finite difference operator σ_{Bq_k}:

$$\tau = \frac{\partial}{\partial q_k} - \sum_{(s,j)\in\sigma_{Bq_k}} sT_{\mathbf{r}_j} \tag{2.23}$$

$$= \mathcal{O}(h_{q_k}). \tag{2.24}$$

This shows the backward finite difference operator to also be of first order.

Central finite difference operator σ_{Cq_k}:

$$\tau = \frac{\partial}{\partial q_k} - \sum_{(s,j)\in\sigma_{Cq_k}} sT_{\mathbf{r}_j} \tag{2.25}$$

$$= \mathcal{O}(h_{q_k}^2). \tag{2.26}$$

This shows the central finite difference operator to be of second order.

"Squared halved central finite difference operator" $\sigma^2_{(Cq_k)^{\frac{1}{2}}}$:

$$\tau = \frac{\partial^2}{\partial q_k^2} - \sum_{(s,j)\in\sigma^2_{(Cq_k)^{\frac{1}{2}}}} sT_{\mathbf{r}_j} \tag{2.27}$$

$$= \frac{\partial^2}{\partial q_k^2} + h_{q_k}^{-2}T_{\mathbf{0}} - h_{q_k}^{-2}T_{-h_{q_k}\mathbf{e}_k} - h_{q_k}^{-2}T_{h_{q_k}\mathbf{e}_k} \tag{2.28}$$

$$= \frac{\partial^2}{\partial q_k^2} + 2h_{q_k}^{-2}\sum_{n=0}^{\infty}\frac{0^n}{n!}\frac{\partial^n}{\partial q_k^n} - h_{q_k}^{-2}\sum_{n=0}^{\infty}\frac{(-h_{q_k})^n}{n!}\frac{\partial^n}{\partial q_k^n} - h_{q_k}^{-2}\sum_{n=0}^{\infty}\frac{h_{q_k}^n}{n!}\frac{\partial^n}{\partial q_k^n} \tag{2.29}$$

$$= \frac{\partial^2}{\partial q_k^2} + 2h_{q_k}^{-2}\mathrm{id} \tag{2.30}$$

$$-h_{q_k}^{-2}\mathrm{id} + h_{q_k}^{-1}\frac{\partial}{\partial q_k} - \frac{1}{2}\frac{\partial^2}{\partial q_k^2} - \frac{h_q}{6}\frac{\partial^3}{\partial q_k^3} - h_{q_k}^{-2}\sum_{n=4}^{\infty}\frac{(-h_{q_k})^n}{n!}\frac{\partial^n}{\partial q_k^n} \tag{2.31}$$

$$-h_{q_k}^{-2}\mathrm{id} - h_{q_k}^{-1}\frac{\partial}{\partial q_k} - \frac{1}{2}\frac{\partial^2}{\partial q_k^2} + \frac{h_q}{6}\frac{\partial^3}{\partial q_k^3} - h_{q_k}^{-2}\sum_{n=4}^{\infty}\frac{h_{q_k}^n}{n!}\frac{\partial^n}{\partial q_k^n} \tag{2.32}$$

$$= -h_{q_k}^{-2}\sum_{n=4}^{\infty}\frac{(-h_{q_k})^n}{n!}\frac{\partial^n}{\partial q_k^n} - h_{q_k}^{-2}\sum_{n=4}^{\infty}\frac{h_{q_k}^n}{n!}\frac{\partial^n}{\partial q_k^n} \tag{2.33}$$

$$= \mathcal{O}(h_{q_k}^2). \tag{2.34}$$

This shows the squared halved central finite difference operator to be of second order.

3. Uniform Traversal

Uniform traversal is done intuitively by traversing through time. Each time layer indicates the subdomains over which the time coordinate is constant:

$\mathcal{T}_n = \{i \in \mathcal{I} \mid \psi(i)_{d+1} = nh_t\}$.

Note that $\mathcal{T}_n$ is a $d_{\mathbf{x}}$-dimensional uniform subdiscretization including lower dimensional boundaries. This system can be solved as a complete system.

Traversing further space dimensions separately might be an option but since spatial domains are defined all around and not one sided, such as the initial conditions do for the time dimension, working from one boundary to the other might run into complications. The end result might not match the other boundary. A self correcting back and forth approach might yield possibilities for solving this problem.

4. Creating the linear systems

4.1. Uniform matrix representation. In 5.1 I introduced the generalized version of the matrix representation.

Q^j

The matrix representation Q^j of the sub atomic finite difference operator ς^j is a matrix such that at row i it is one exactly at the column position of j-th neighbor of i and zero elsewhere.

For uniform discretizations the matrix representation will consist of punctured banded systems. The punctures in these band are formed by the fact that at the start or end of 'row' the node with the next respectively previous index is not a neighbor. It has 'jumped' to the beginning of next 'row' or to the end of the previous 'row'. For higher dimensions these gaps become larger as not a one dimensional 'row' is skipped but a two dimensional 'slice', a three dimensional 'volume' etcetera. Cf. Figure 1.

For uniform systems these punctured banded systems can be described using Kronecker products of matrices:

$$Q^{\mathbf{j}}_{\mathcal{T}_n,\mathcal{T}_{n-p}} := \bigotimes_{k=1}^{d} J^{j_k}_{m_k} \tag{4.1}$$

$$:= J^{j_1}_{m_1} \otimes \ldots \otimes J^{j_d}_{m_d} \qquad \forall \mathbf{j} \in \mathbb{Z}^d \times \{-p\} \tag{4.2}$$

To prove that (4.1) is indeed the matrix representation belonging to the uniform discretization of definition 16 I proof the following lemma.

LEMMA 4. *The matrix representation of the uniform discretization of definition 16 is indeed given by:*

$$Q^{\mathbf{j}}_{\mathcal{T}_n,\mathcal{T}_{n-p}} := \bigotimes_{k=1}^{d} J^{j_k}_{m_k} \tag{4.3}$$

as stated in definition 5.5.

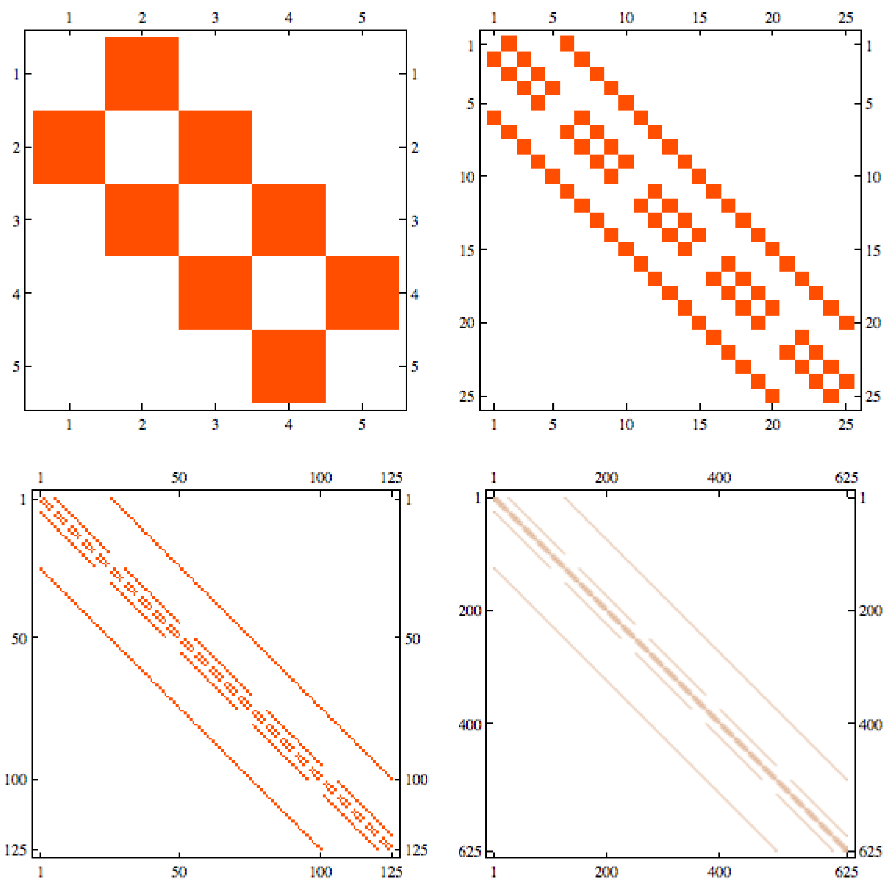


FIGURE 1. Example of the punctured banded systems arising from the Kronecker sequences for $d = 1, 2, 3, 4$ for $S = \sum_{k=1}^{d} \sigma_{Cx_k}$. with $\mathbf{m} = \mathbf{5}$.

Proof:

$$(Q^{\mathbf{j}}_{\mathcal{T}_n,\mathcal{T}_n})_{i,g} := (\bigotimes_{k=1}^{d} J^{j_k}_{m_k})_{i,g} \tag{4.4}$$

$$= \left(\bigotimes_{k=2}^{d} J^{j_k}_{m_k} \otimes \left(\underset{i_1=1}{\overset{m_1}{\boxed{\rightarrow}}} \underset{g_1=0}{\overset{m_1}{\boxed{\downarrow}}} \delta_{i_1,g_1+j_1} \right) \right)_{i,g} . \tag{4.5}$$

For this value to be one the first criterium is that δ_{i_1,g_1+j_1} must be one. This is equivalent to:

$$
\begin{aligned}
& \delta_{i_1,g_1+j_1} = 1 \\
\Leftrightarrow\quad & i_1 = g_1 + j_1 \\
\Leftrightarrow\quad & i + am_1 = g + bm_1 + j_1 \quad \forall a,b \in \{1,\dots,\prod_{k'=2}^{d} m_{k'}\} \\
\Leftrightarrow\quad & i + (a-b)m_1 = g + j_1 \quad \forall a,b \in \{1,\dots,\prod_{k'=2}^{d} m_{k'}\} \\
\Rightarrow\quad & i \bmod m_1 = g + j_1.
\end{aligned}
\tag{4.6}
$$

From this follows that

$$
= \left(\bigotimes_{k=2}^{d} J_{m_k}^{j_k} \otimes \left(\overset{m_1}{\underset{i_1=1}{\boxed{\rightarrow}}} \overset{m_1}{\underset{g_1=0}{\boxed{\downarrow}}} \; 1 \right) \right)_{i,g} \delta_{i \bmod m_1, g+j_1} \tag{4.7}
$$

$$
= \left(\bigotimes_{k=2}^{d} J_{m_k}^{j_k} \otimes \mathbf{1}_{m_1,m_1} \right)_{i,g} \delta_{i \bmod m_1, g+j_1}, \tag{4.8}
$$

where $\mathbf{1}_{m,m}$ is the $m \times m$-matrix with all ones. We can repeat this process another $d-1$ times by stating that:

$$
\begin{aligned}
& \delta_{i_k,g_k+j_k} \\
\Leftrightarrow\quad & i_k = g_k + j_1 \\
\Leftrightarrow\quad & \left\lfloor \frac{i}{\mathbf{m}^{k-1}} \right\rfloor + am_k = g + bm_k + j_1 \quad \forall a,b \in \{1,\dots,\prod_{k'=k+1}^{d} m_{k'}\} \\
\Leftrightarrow\quad & \left\lfloor \frac{i}{\mathbf{m}^{k-1}} \right\rfloor + (a-b)m_k = g + j_1 \quad \forall a,b \in \{1,\dots,\prod_{k'=k+1}^{d} m_{k'}\} \\
\Rightarrow\quad & \left\lfloor \frac{i}{\mathbf{m}^{k-1}} \right\rfloor \bmod m_k = g + j_1,
\end{aligned}
\tag{4.9}
$$

yielding:

$$
= \left(\bigotimes_{k=1}^{d} \mathbf{1}_{m_k,m_k} \right)_{i,g} \prod_{k=1}^{d} \delta_{\left\lfloor \frac{i}{\mathbf{m}^{k-1}} \right\rfloor \bmod m_k, g+j_k} \tag{4.10}
$$

$$
= \left(\mathbf{1}_{\mathbf{m}^d,\mathbf{m}^d} \right)_{i,g} \prod_{k=1}^{d} \delta_{\left\lfloor \frac{i}{\mathbf{m}^{k-1}} \right\rfloor \bmod m_k, g+j_k} \tag{4.11}
$$

$$
= \prod_{k=1}^{d} \delta_{\left\lfloor \frac{i}{\mathbf{m}^{k-1}} \right\rfloor \bmod m_k, g+j_k}. \tag{4.12}
$$

These d equivalencies are solved by the following equivalency

$$= \delta_{i,g+\sum_{k=1}^{d} j_k \mathbf{m}^{k-1}}, \tag{4.13}$$

Which is identical to applying the adjacency operator:

$$= \delta_{i,g+\Delta \mathbf{j}}. \tag{4.14}$$

this conclude the proof since it conforms to the definition of Q in 5.5 □

4.2. Boundaries. For the boundary term, contributions are described by $Q^{\mathbf{j}}_{\mathcal{T}_n,\partial_{k,l}\mathcal{T}_n}$. For $k = d$ this is equal to the already defined internal matrix representation. For $k < d$ it follows that $\partial_{k,l}\mathcal{T}_n \bigcap \mathcal{T}_n = \emptyset$. The identity or zero label $\mathbf{j} = \mathbf{0}$ will thereforealways yield a zero matrix. (The boundary will contain nodes left, right, etc., of the internal nodes but never the internal nodes themselves.)

$$Q^{\mathbf{j}}_{\mathcal{T}_n,\partial_{k,l}\mathcal{T}_n} := \begin{cases} \mathbf{0} & k < d \wedge \mathbf{j} = \mathbf{0}, \\ \bigotimes_{k'=1}^{d} \left(\begin{cases} J^{j_{k'}}_{m_{k'}} & k' \in \text{perm}(\left\lfloor \frac{l}{2^{d-k}} \right\rfloor, d-k, d) \\ \partial J^{j_{k'}}_{m_{k'}} & k' \notin \text{perm}(\left\lfloor \frac{l}{2^{d-k}} \right\rfloor, d-k, d) \end{cases} \right) & \text{otherwise.} \end{cases}$$

Here $\text{perm}(a, b, c)$ is the a-th permutation of b numbers from numbers $1, \ldots, c$ and $\partial J^{j_{k'}}_{m_{k'}} = \mathbf{e}_{j_{k'} \bmod m_{k'}}$. The $J^{j_{k'}}_{m_{k'}}$ represent the lower dimensional boundaries and $\partial J^{j_{k'}}_{m_{k'}}$ the restrictions layer upon this lower dimensional region. The two dimensional unit square has four one dimensional boundaries: the north edge is the one dimensional domain restricted by $y = 1$, the south by $y = -1$ and east and west by $x = -1$ and $x = 1$ respectively. The four zero dimensional corners all have two restrictions: for example northwest is the intersection of $x = 1$ and $y = 1$.

5. Solving the Linear System

When computing the solution to a linear system in practice one seldom computes the explicit inverse of a matrix. Similarly matrices which are the result of the Kronecker sequences need not be computed explicitly. This would not only be a time intensive but also memory intensive process. The solution methods are applied to the three test cases (1 Heat, 2 Implosion, 3 Burgers') which where introduced in sections 3.1, 3.2 and 3.3.

The multiplications are computed using several BLAS subroutines, with these functions a BiCGStab solver [**4**]. was constructed.

The local truncation error for each FTCS and BTCS discretization of the test cases is $\mathcal{O}(h_t) + \mathcal{O}(h_x^2)$. De derivation of the local truncation errors and the stability criteria are detailed in Appendix 5.

Results for the uniform cases are included in the method comparison of part 4. See section 6.5 for the results and section 6.5 for considerations on dimensional comparison.

Test case	Discretization	Stability
1	FTCS	Conditionally stable
1	BTCS	Unconditionally stable
2	FTCS	Unconditionally unstable
2	BTCS	Unconditionally stable
3	FTCS	Conditionally stable
3	BTCS	Unconditionally stable

6. Uniform Conclusions

The uniform domain discretization yields several favorable properties:

Advantages:

- Easy and efficient sparse matrix operations
- Easy operator operations (Higher order derivatives)
- (Relatively) easy analysis: truncation error and stability estimates

Disadvantages:

- Local truncation and stability issues can only be resolved globally. Resulting in memory and or computationally intensive improvements.

CHAPTER 4

d-dimensional Adaptive FDM

1. Introduction

The exponential increase of resources needed to solve higher dimensional problems poses a serious limitation on implementations. Fundamental improvement are needed to extend the method to higher dimensions. The uniform method described in the previous Part 3 has several options for improvement which, ideally, would retain its advantages:

1 Improve the domain discretization

- 1a Decrease h_t
  Can only reduce the $\mathcal{O}(h_t^k)$ not the $\mathcal{O}(h_x^k)$ terms of the local truncation error.
- 1b Decrease h_x globally
  Reducing the node distance by increasing the node density will increase the number of nodes and the memory footprint. The increase is exponential with dimension.
- 1c Decrease h_x locally
  This only adapts the node densities in regions perpendicular to the axis or voids the uniform structure.
- 1d h-refinement : add/remove nodes at strategic positions.
  This voids the uniform structure.

2 Improve operator discretization

- 2a Use a higher order approximation globally
  Definitely an option, but not really a method improvement.
- 2b p-refinement : Use a higher order approximation at strategic positions
  This voids the uniform structure. Only when using separate traversal levels for separate approximations could this be an option but then simultaneous solving of traversal levels should be implemented.

3 Improve the traversal
As mentioned in section 5.3 spatial traversal has its difficulties but for special cases it might be an option.

4 r-refinement Transform the problem itself
By transforming the problem into two separate problems: one for adapted node coordinates and one for solving the solution over these coordinates. The first is done in a matter that is beneficial to the solving of the second.

For this thesis I will apply an r-refinement by transforming the differential equation. In a way this is more a case of adapting the problem to the structure than the structure to the problem since the structure was already favorably to the finite difference method by using a uniform domain discretization. The idea is to dissect the solution function $u(\mathbf{q})$ into a function composition $v \circ \mathbf{q}(\koppa)$. This way coordinates $\mathbf{q}$ are themselves viewed as a function of original uniform coordinates $\koppa$. ($\koppa$ or *koppa* being a - uncommon - Greek character for q as ξ and θ are for x respectively t.)The grid equation: a boundary value problem (BVP) is stated to solve for these adapted coordinates. The resulting adapted space-time coordinates will be denoted by $\mathbf{q} = (\mathbf{x}, t)$ and the original uniform by $\koppa = (\boldsymbol{\xi}, \theta)$.

These adapted coordinates are then used to solve for v. The transformation of u into v is not done explicitly, instead it is the differential operator L that (partly) defines the PDE that is transformed.

To benefit from this transformation a grid equation is formulated such that it would yield a solution that is favorable to the solving of v. In this thesis two methods are explored: the method of Characteristics[**5**] which can only be applied to certain cases (Test case II) and the more general Winslow method [**6**]. With these node positions a the transformation of the original boundary value problem is solved. Both BVPs are solved using the uniform finite difference solution methods presented in earlier sections. While the equation is transformed the discretization of the solution function remains the same:

$$\mathbf{v}_i := v(\mathbf{q}_i) := v(\mathbf{q}(\koppa_i)) = (v \circ \mathbf{q})(\koppa_i) =: u(\koppa_i) =: \mathbf{u}_i. \tag{1.1}$$

2. Monitor functions

A monitor function assigns priority to the node density in a given region. The higher the monitor function at a certain point the higher the node density should be. The absolute value of first order derivative is often a good indicator for the ‘turbulence’ and thus the need for more nodes to properly represent the approximation. The summation of all absolute first order spatial derivatives can be denoted as:

$$\omega = ||\nabla u||^2. \tag{2.1}$$

Using the considerations of [**8**] a new floor value $\alpha(u)$ is included as the average gradient, resulting in:

$$\omega = \alpha(u) + ||\nabla u||^2, \tag{2.2}$$

with $\alpha(u) = \frac{\int_\Omega ||\nabla u|| \, \mathrm{d}\mathbf{q}}{\int_\Omega 1 \, \mathrm{d}\mathbf{q}}$.

3. Transformations

3.1. First order. A first order derivative is transformed by:

$$\begin{aligned}\frac{\partial}{\partial q_k} &= \sum_{l=1}^{d_q} \frac{\partial}{\partial \natural_l}\frac{\partial \natural_l}{\partial q_k} \\ &= \sum_{l=1}^{d_q} \frac{\partial}{\partial \natural_l}\mathcal{J}_{k,l}^{-1}\end{aligned} \tag{3.1}$$

$\frac{\partial}{\partial \natural_l}$ does not operate on but is scaled by $\mathcal{J}_{k,l}^{-1}$. To clarify that it remains an operator and not an operator working on the inverse Jacobian, exchange order such that:

$$= \sum_{l=1}^{d_q} \mathcal{J}_{k,l}^{-1}\frac{\partial}{\partial \natural_l}.$$

3.2. Second order. An arbitrary mixed second order derivative is transformed as follows:

$$\frac{\partial^2}{\partial q_{k_1} q_{k_2}} = \frac{\partial}{\partial q_{k_1}}\left(\frac{\partial}{\partial q_{k_2}}\right). \tag{3.2}$$

Apply (3.1):

$$= \frac{\partial}{\partial q_{k_1}}\left(\sum_{l=1}^{d_q} \mathcal{J}_{k_2,l}^{-1}\frac{\partial}{\partial \natural_l}\right).$$

Apply product rule:

$$= \sum_{l=1}^{d_q} \frac{\partial}{\partial q_{k_1}}\mathcal{J}_{k_2,l}^{-1}\frac{\partial}{\partial \natural_l} + \mathcal{J}_{k_2,l}^{-1}\frac{\partial}{\partial q_{k_1}}\frac{\partial}{\partial \natural_l}$$

Apply (3.1):

$$\begin{aligned}&= \sum_{l=1}^{d_q}\left(\sum_{m=1}^{d_q} \mathcal{J}_{k_1,m}^{-1}\frac{\partial}{\partial \natural_m}\right)\mathcal{J}_{k_2,l}^{-1}\frac{\partial}{\partial \natural_l} + \mathcal{J}_{k_2,l}^{-1}\left(\sum_{m=1}^{d_q} \mathcal{J}_{k_1,m}^{-1}\frac{\partial}{\partial \natural_m}\right)\frac{\partial}{\partial \natural_l}. \\ &= \sum_{l=1}^{d_q}\sum_{m=1}^{d_q} \mathcal{J}_{k_1,m}^{-1}\frac{\partial \mathcal{J}_{k_2,l}^{-1}}{\partial \natural_m}\frac{\partial}{\partial \natural_l} + \mathcal{J}_{k_2,l}^{-1}\mathcal{J}_{k_1,m}^{-1}\frac{\partial^2}{\partial \natural_m \natural_l}.\end{aligned}$$

3.3. Higher order. An arbitrary mixed n-th order derivative is transformed as follows:

$$\bigcirc_{j=1}^{n} \frac{\partial}{\partial x_{k_j}} = \bigcirc_{j=1}^{n-1}\left(\frac{\partial}{\partial \natural_{k_j}} \circ \frac{\partial \natural_{k_j}}{\partial x_{k_n}}\right). \tag{3.3}$$

I did not succeed in deriving a more explicit form. It should be remarked that the number of terms grows exponentially both with increase of dimensions and with increase of derivative order resulting in vast systems. This is a first doubt for the fitness of this approach for higher-order higher-dimensional problems.

4. Testcase 2: Implosion equation

4.1. One dimensional example. Consider the PDE of test case 2:

$$u_t = xu_x. \tag{4.1}$$

The value of u at the boundary which in the analytical case tends to zero towards infinity, is forced to zero at a closer (finite) distance which simplifies matters.

Equation (4.1) yields the differential operator defined as L_u, the transformation of the x coordinate will be solved by using another differential operator L_x

$$L_u = \frac{\partial}{\partial t} - x\frac{\partial}{\partial x}. \tag{4.2}$$

Note that x is a (trivial) function of x so we can denote

$$L_u = \frac{\partial}{\partial t} - x(\cdot)\frac{\partial}{\partial x}. \tag{4.3}$$

Discretizing L_u with finite difference operator S_u yields:

$$S_u = S_{(\cdot)t} - \operatorname{diag}(\mathbf{x})S_{(\cdot)x}, \tag{4.4}$$

where $(\cdot)$ denotes the freedom to choose a finite difference stencil (forward, backward, etc) .

To use an adaptive grid method define x and t as functions of new uniform variables ξ, θ and define $v(\xi, \theta) = u(x(\xi, \theta), t(\xi, \theta))$.

Using the following identities:

$$\frac{\partial v}{\partial x} = \frac{\partial v}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial v}{\partial \theta}\frac{\partial \theta}{\partial x} \tag{4.5}$$

$$\frac{\partial v}{\partial t} = \frac{\partial v}{\partial \xi}\frac{\partial \xi}{\partial t} + \frac{\partial v}{\partial \theta}\frac{\partial \theta}{\partial t}. \tag{4.6}$$

The PDE can be transformed:

$$\frac{\partial v}{\partial \xi}\frac{\partial \xi}{\partial t} + \frac{\partial v}{\partial \theta}\frac{\partial \theta}{\partial t} = x(\cdot)\left(\frac{\partial v}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial v}{\partial \theta}\frac{\partial \theta}{\partial x}\right). \tag{4.7}$$

(4.8)

Note that x is no longer the trivial function but a function of variable ξ and θ : $x(\xi, \theta)$.

$$L_v = \frac{\partial \xi}{\partial t}\frac{\partial}{\partial \xi} + \frac{\partial \theta}{\partial t}\frac{\partial}{\partial \theta} - \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} - \frac{\partial \theta}{\partial x}\frac{\partial}{\partial \theta}. \tag{4.9}$$

The derivatives of ξ and θ can be determined using the matrix equation for the Jacobian, $\mathcal{J}$ matrix and its inverse $\mathcal{J}^{-1}$:

$$\mathcal{J}\mathcal{J}^{-1} = I \tag{4.10}$$

$$\Leftrightarrow \begin{pmatrix} \frac{\partial\xi}{\partial x} & \frac{\partial\xi}{\partial t} \\ \frac{\partial\theta}{\partial x} & \frac{\partial\theta}{\partial t} \end{pmatrix} \begin{pmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial x}{\partial\theta} \\ \frac{\partial t}{\partial\xi} & \frac{\partial t}{\partial\theta} \end{pmatrix} = I. \tag{4.11}$$

Note that, for now, I have chosen $t \equiv \theta$. The inverse of the Jacobian is now computed as:

$$\begin{pmatrix} \frac{\partial\xi}{\partial x} & \frac{\partial\xi}{\partial t} \\ \frac{\partial\theta}{\partial x} & \frac{\partial\theta}{\partial t} \end{pmatrix} \begin{pmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial x}{\partial\theta} \\ 0 & 1 \end{pmatrix} = I \tag{4.12}$$

$$\begin{pmatrix} \frac{\partial\xi}{\partial x} & \frac{\partial\xi}{\partial t} \\ \frac{\partial\theta}{\partial x} & \frac{\partial\theta}{\partial t} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial\xi} & \frac{\partial x}{\partial\theta} \\ 0 & 1 \end{pmatrix}^{-1} \tag{4.13}$$

$$= \begin{pmatrix} \frac{1}{\frac{\partial x}{\partial\xi}} & -\frac{\frac{\partial x}{\partial\theta}}{\frac{\partial x}{\partial\xi}} \\ 0 & 1 \end{pmatrix}. \tag{4.14}$$

The transformed differential operator can be written as:

$$L_v := \left(\left(-\frac{\frac{\partial x}{\partial\theta}}{\frac{\partial x}{\partial\xi}}\right)\frac{\partial}{\partial\xi} + 1\,\frac{\partial}{\partial\theta}\right) - x\left(\left(\frac{1}{\frac{\partial x}{\partial\xi}}\right)\frac{\partial}{\partial\xi} + 0\,\frac{\partial}{\partial\theta}\right) = 0. \tag{4.15}$$

This gives:

$$L_v = \frac{\partial}{\partial\theta} - \left(\frac{x + x_\theta}{x_\xi}\right)\frac{\partial}{\partial\xi} \tag{4.16}$$

$$= \frac{\partial}{\partial\theta} - \left(\frac{\mathrm{id} + \frac{\partial}{\partial\theta}}{\frac{\partial}{\partial\xi}}\right) x \frac{\partial}{\partial\xi} \tag{4.17}$$

Note that $L_x := \frac{\mathrm{id}+\frac{\partial}{\partial\theta}}{\frac{\partial}{\partial\xi}}$ is an operator that operates on x while the complete operator operates on v.

$$= \frac{\partial}{\partial\theta} - L_x x\,\frac{\partial}{\partial\xi}. \tag{4.18}$$

This nested structure is used when observing that when $L_x = 0$ it follows that $L_u = \frac{\partial}{\partial\theta}$ and the equation thereforebecomes: $\frac{\partial}{\partial\theta} = 0$. This results in a solution that is constant over time and thus equal to the initial condition. This is the basis for the *method of Characteristics.* [**5**] A powerful method to solve a PDE, if the equation permits it.

Discretization gives:

$$S_v := S_{(\cdot)\theta} - \mathrm{diag}\left(S_x\mathbf{x}\right) S_{(\cdot)\xi}, \tag{4.19}$$

with $S_x = \frac{I+S_{(\cdot)\theta}}{S_{(\cdot)\xi}}$. To implement S_v discretization choices for each $(\cdot)$ must be made. As an example the derivation of the complete matrix representation using BTCS (Backwards in time, central in space) discretization:

$$S_v^{BTCS} := S_{B\theta} - \operatorname{diag}\left(S_x^{BTCS}\mathbf{x}\right) S_{C\xi}, \tag{4.20}$$

with $S_x^{BTCS} = \frac{I+S_{B\theta}}{S_{C\xi}}$.

Since x must cover the domain, its boundary values are $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$, and these cannot be forced to zero.

In stencil form:
(4.21)

$$S_v^{BTCS} = \left\{(h_\theta^{-1}, \mathbf{0}), (-h_\theta^{-1}, -\mathbf{e}_2), (-(2h_\xi)^{-1}\operatorname{diag}\left(S_v^{BTCS}\mathbf{x}\right), \mathbf{e}_1), ((2h_\xi)^{-1}\operatorname{diag}\left(S_v^{BTCS}\mathbf{x}\right), -\mathbf{e}_1)\right\},$$

with $S_x^{BTCS} = \frac{\{(1,\mathbf{0}),(h_\theta^{-1},\mathbf{0}),(-h_\theta^{-1},-\mathbf{e}_2)\}}{\{((2h_\xi)^{-1},\mathbf{e}_2),(-(2h_\xi)^{-1},-\mathbf{e}_2)\}}$.

In matrix form:

$$\begin{aligned}
S_v^{BTCS} &= h_\theta^{-1}Q_{\mathbf{0}} - h_\theta^{-1}Q_{-\mathbf{e}_2} - (2h_\xi)^{-1}\operatorname{diag}\left(S_x^{BTCS}\mathbf{x}\right)Q_{\mathbf{e}_1} + (2h_\xi)^{-1}\operatorname{diag}\left(S_x^{BTCS}\mathbf{x}\right)Q_{-\mathbf{e}_1}\\
&= h_\theta^{-1}\left(Q_{\mathbf{0}} - Q_{-\mathbf{e}_2}\right) - (2h_\xi)^{-1}\operatorname{diag}\left(S_x^{BTCS}\mathbf{x}\right)\left(Q_{\mathbf{e}_1} - Q_{-\mathbf{e}_1}\right)\\
&= h_\theta^{-1}\left(J^0\otimes J^0 - J^0\otimes J^{-1}\right) - (2h_\xi)^{-1}\operatorname{diag}\left(S_x^{BTCS}\mathbf{x}\right)\left(J^1\otimes J^0 - J^{-1}\otimes J^0\right)\\
&= h_\theta^{-1}\left(I - I\otimes J^T\right) - (2h_\xi)^{-1}\operatorname{diag}\left(S_x^{BTCS}\mathbf{x}\right)\left(J\otimes I - J^T\otimes I\right),
\end{aligned}$$

with $S_x^{BTCS} = \frac{I+h_\theta^{-1}\left(Q_{\mathbf{0}}-Q_{-\mathbf{e}_2}\right)}{(2h_\xi)^{-1}\left(Q_{\mathbf{e}_1}-Q_{-\mathbf{e}_1}\right)} = \frac{I+h_\theta^{-1}\left(J^0\otimes J^0-J^0\otimes J^{-1}\right)}{(2h_\xi)^{-1}(J^1\otimes J^0-J^{-1}\otimes J^0)} = \frac{I+h_\theta^{-1}\left(I-I\otimes J^T\right)}{(2h_\xi)^{-1}(J\otimes I-J^T\otimes I)}$.

Applied to solution vector $\mathbf{v}$ per time level n:

$$\left(S_{v,BTCS}\mathbf{v}\right)^n = h_\theta^{-1}\left(\mathbf{v}^n - \mathbf{v}^{n-1}\right) - (2h_\xi)^{-1}\operatorname{diag}\left(\frac{\mathbf{x}^n + h_t^{-1}\left(\mathbf{x}^n - \mathbf{x}^{n-1}\right)}{(2h_x)^{-1}\left(J\mathbf{x}^n - J^T\mathbf{x}^n\right)}\right)\left(J\mathbf{v}^n - J^T\mathbf{v}^n\right)\forall n$$

Since $\mathbf{v}_i = v(ϟ_i) = u(\mathbf{q}(ϟ_i)) = u(\mathbf{q}_i) = \mathbf{u}_i$ it follows that $\mathbf{u} = \mathbf{v}$. $\mathbf{u}$ and $\mathbf{v}$ are therefore interchangeable and $\mathbf{u}$ will be used from now on.

To construct an equation for $\mathbf{u}^n$ assume that $\mathbf{x}^n, \mathbf{x}^{n-1}$ and $\mathbf{u}^{n-1}$ known. A nodal representation for node i at time level n is given by:

$$\left(S_v\mathbf{u}^n\right)_i = \left(\frac{\mathbf{x}_i^n - \frac{\mathbf{x}_i^n - \mathbf{x}_i^{n-1}}{h_\theta}}{\frac{\mathbf{x}_{i+1}^n - \mathbf{x}_{i-1}^n}{2h_\xi}}\right)\frac{\mathbf{u}_{i+1}^n - \mathbf{u}_{i-1}^n}{2h_\xi} + \frac{\mathbf{u}_i^n - \mathbf{u}_i^{n-1}}{h_\theta}. \tag{4.22}$$

This explicated form is only mentioned to connect to classical per node formulation which are renowned to become overly complicated for higher dimensional problems and for problems with non zero boundary conditions. Describing the discretization by finite difference operators remain compact even for higher dimensions and with boundary terms. On top of that it provides flexibility to choose the method of discretization.

4.2. Higher dimensional example.

$$u_t = \mathbf{x} \cdot \nabla u \tag{4.23}$$

from which we extract the differential operator:

$$L_u = \frac{\partial}{\partial t} - \mathbf{x}(\cdot) \cdot \nabla \tag{4.24}$$

$$= \frac{\partial}{\partial t} - \sum_{k=1}^{d} x_k(\cdot) \frac{\partial}{\partial x_k}, \tag{4.25}$$

which is, using generalized space time coordinates $\mathbf{q} := (q_1, \dots, q_d, q_{d+1}) = (x_1, \dots, x_d, t)$, equivalent to

$$L_u = \frac{\partial}{\partial q_{d+1}} - \sum_{k=1}^{d} q_k(\cdot) \frac{\partial}{\partial q_k}. \tag{4.26}$$

Analogous to the definition of the generalized coordinates I introduce the generalized adaptive space time coordinates.

$$\daleth := (\daleth_1, \dots, \daleth_d, \daleth_{d+1}) = (\xi_1, \dots, \xi_d, \theta). \tag{4.27}$$

Define each variable q_k to be function of adaptive variables $\daleth_1, \dots, \daleth_{d+1} : q_k(\daleth_1, \dots, \daleth_{d+1})$ and use the following identity to transform the equation:

$$\frac{\partial u}{\partial q_i} = \sum_{k=1}^{d+1} \frac{\partial u}{\partial \daleth_k} \frac{\partial \daleth_k}{\partial q_i}. \tag{4.28}$$

The Jacobian is used to compute the inverse transformation:

$$\mathcal{J}\mathcal{J}^{-1} = I \tag{4.29}$$

$$\Leftrightarrow \begin{pmatrix} \frac{\partial \daleth_1}{\partial q_1} & \cdots & \frac{\partial \daleth_1}{\partial q_{d+1}} \\ \vdots & & \vdots \\ \frac{\partial \daleth_{d+1}}{\partial q_1} & \cdots & \frac{\partial \daleth_{d+1}}{\partial q_{d+1}} \end{pmatrix} \begin{pmatrix} \frac{\partial q_1}{\partial \daleth_1} & \cdots & \frac{\partial q_1}{\partial \daleth_{d+1}} \\ \vdots & & \vdots \\ \frac{\partial q_{d+1}}{\partial \daleth_1} & \cdots & \frac{\partial q_{d+1}}{\partial \daleth_{d+1}} \end{pmatrix} = I, \tag{4.30}$$

such that

$$\frac{\partial u}{\partial q_i} = \sum_{k=1}^{d+1} (\mathcal{J}^{-1})_{i,k} \frac{\partial u}{\partial \daleth_k}. \tag{4.31}$$

Now (4.26) can be transformed into adaptive form:

$$L_v = \sum_{k=1}^{d+1} (\mathcal{J}^{-1})_{d+1,k} \frac{\partial}{\partial \daleth_k} - \sum_{k=1}^{d} q_k \sum_{k'=1}^{d+1} (\mathcal{J}^{-1})_{k,k'} \frac{\partial}{\partial \daleth_{k'}} = 0. \tag{4.32}$$

Note that unlike the one dimensional case I have not yet explicitly computed $\mathcal{J}^{-1}$. In its symbolical form the Jacobian matrix is a $(d+1) \times (d+1)$ matrix which is, for small d perfectly within the scope of direct solution methods (i.e. Guassian elimination or

Cramers' rule). instead of computing $\mathcal{J}^{-1}$ symbolically with the possibilities of terms coinciding or canceling each other I chose to discretize $\mathcal{J}$ into $\mathbf{J}$, a matrix of diagonal matrices (a 'matrix-matrix'), for which addition, multiplication and division operations are conceptually equivalent to scalars in a regular scalar-matrix.

$$\mathbf{J} = \begin{pmatrix} \operatorname{diag}(S_{(\cdot)\natural_1}\mathbf{q}_1) & \dots & \operatorname{diag}(S_{(\cdot)\natural_{d+1}}\mathbf{q}_1) \\ \vdots & & \vdots \\ \operatorname{diag}(S_{(\cdot)\natural_1}\mathbf{q}_{d+1}) & \dots & \operatorname{diag}(S_{(\cdot)\natural_{d+1}}\mathbf{q}_{d+1}) \end{pmatrix}. \tag{4.33}$$

This choice was made to contain the scope of the method to a more numerical but it does give options for further improvements. For the inversion of the Jacobian I've implemented Cramers' rule on the matrix-matrix and BLAS QLUP decomposition per node i . Both with similar results. But the determinant for Cramers' rule was computed in a naive way so is open to further improvement. A third option: transforming the matrix-matrix matrix equation to a matrix vector equation and applying BiCGStab was also tried but proved much less efficient in its rough application so it was not explored any further.

$$\begin{aligned} (\mathbf{J}_{BTCS})^n &= \begin{pmatrix} \operatorname{diag}((S_{C\natural_1}\mathbf{q}_1)^n) & \dots & \operatorname{diag}((S_{C\natural_d}\mathbf{q}_1)^n) & \operatorname{diag}((S_{B\natural_{d+1}}\mathbf{q}_d)^n) \\ \vdots & & \vdots & \vdots \\ \operatorname{diag}((S_{C\natural_1}\mathbf{q}_d)^n) & \dots & \operatorname{diag}((S_{C\natural_{d+1}}\mathbf{q}_{d+1})^n) & \operatorname{diag}((S_{B\natural_{d+1}}\mathbf{q}_{d+1})^n) \end{pmatrix} \\ &= \begin{pmatrix} \operatorname{diag}(\frac{Q_{\mathbf{e}_1}-Q_{-\mathbf{e}_1}}{2h_{q_1}}\mathbf{q}_1^n) & \dots & \operatorname{diag}(\frac{Q_{\mathbf{e}_d}-Q_{-\mathbf{e}_d}}{2h_{q_d}}\mathbf{q}_1^n) & \operatorname{diag}(\frac{\mathbf{q}_1^n-\mathbf{q}_1^{n-1}}{h_{q_{d+1}}}) \\ \vdots & & \vdots & \vdots \\ \operatorname{diag}(\frac{Q_{\mathbf{e}_1}-Q_{-\mathbf{e}_1}}{2h_{q_1}}\mathbf{q}_{d+1}^n) & \dots & \operatorname{diag}(\frac{Q_{\mathbf{e}_d}-Q_{-\mathbf{e}_d}}{2h_{q_d}}\mathbf{q}_{d+1}^n) & \operatorname{diag}(\frac{\mathbf{q}_{d+1}^n-\mathbf{q}_{d+1}^{n-1}}{h_{q_{d+1}}}) \end{pmatrix}. \end{aligned}$$

The boundary terms for $\mathbf{q}_k$ cannot be discarded as they cannot be forced to zero on all boundaries. Their boundary terms are dictated by $\mathbf{q}_{\max}$ and $\mathbf{q}_{\min}$.

Using the system of linear equations in diagonal matrices at each step n

$$\mathbf{J}^n(\mathbf{J}^{-1})^n \;=\; I_I, \tag{4.34}$$

where I_I is a diagonal matrix with identity matrices as values. Since $\mathbf{v}_i = v(\natural_i) = u(\mathbf{q}(\natural_i)) = u(\mathbf{q}_i) = \mathbf{u}_i$ it follows that $\mathbf{u} = \mathbf{v}$. $\mathbf{u}$ and $\mathbf{v}$ are therefore interchangeable and u will be used from now on.

Use the discretized Jacobi matrices to formulate a uniform grid equation:

$$\sum_{k=1}^{d} S_{(\cdot)\natural_k} \operatorname{diag}\left((\mathbf{J}^{-1})_{d+1,k}\right)\mathbf{u} = \sum_{k=1}^{d} \operatorname{diag}(\mathbf{x}_k) \sum_{k'=1}^{d} S_{(\cdot)\natural_{k'}} \operatorname{diag}\left((\mathbf{J}^{-1})_{k,k'}\right)\mathbf{u}. \tag{4.35}$$

These consist only of constant banded Kronecker sequence matrices and diagonal matrices. All very computational and memory efficient. Note that even the fractions, needed to compute the inverse of the Jacobian via Cramers' rule ,can be computed using the inverses of the diagonal matrix, which are again diagonal matrices with per element inversion. This provides the tools to implement adaptive grids to a very wide variety of problems

5. Method of Characteristics

The Method of Characteristics [**5**] is based on transforming L_u into a operator that yields an ordinary differential equation (ODE). Observing the differential operator of (4.16):

$$L_v = \frac{\partial}{\partial\theta} - \left(\frac{x + x_\theta}{x_\xi}\right)\frac{\partial}{\partial\xi}$$

a promising choice for an adapted grid is one such that $x = x_\theta$. If this is the case the main equation (4.1) reduces to:

$$u_\theta = 0. \tag{5.1}$$

The solution of which remains constant over time thus is equal to the initial condition for u.

5.1. One dimensional case.

Using the Method of Characteristics we have the equation

$$x = -x_\theta \tag{5.2}$$

with differential operator

$$L_{x(\xi,\theta)} := \mathrm{id} + \frac{\partial}{\partial\theta}. \tag{5.3}$$

BT discretization gives:

$$S_{x(\xi,\theta)} := I + \sigma_{B\theta}. \tag{5.4}$$

The finite difference equation is given by:

$$S_{x(\xi,\theta)}\mathbf{x} = 0 \tag{5.5}$$

$$\Leftrightarrow \quad \mathbf{x}_i^n + \frac{\mathbf{x}_i^n - \mathbf{x}_i^{n-1}}{\Delta\theta} = 0 \qquad \forall n, i, \tag{5.6}$$

$$\Leftrightarrow \quad \mathbf{x}_i^n = \frac{1}{\Delta\theta + 1}\mathbf{x}_i^{n-1}. \tag{5.7}$$

Solving this will result in a one-dimensional grid that adapts to maintain a constant solution since $\frac{\partial u}{\partial\theta} = 0$ for these nodes.

5.2. d-dimensional case.

Using Method of Characteristics we have the d differential equations:

$$\mathbf{x} = -\mathbf{x}_\theta, \tag{5.8}$$

each with differential operator

$$L_{x(\xi,\theta)} := \mathrm{id} + \frac{\partial}{\partial\theta}, \tag{5.9}$$

which is discretized exactly as the one dimensional case resulting in d finite difference equations:

$$S_{x(\xi,\theta)}\mathbf{x} = 0 \tag{5.10}$$

$$\Leftrightarrow \quad (\mathbf{x}_k)_i^n = \frac{1}{\Delta\theta + 1}(\mathbf{x}_k)_i^{n-1}, \quad \forall n, i. \tag{5.11}$$

Solving these equations will result in a d-dimensional grid that adapts to maintain a constant solution since $\frac{\partial u}{\partial \theta} = 0$ for these nodes.

6. Winslow Method

Contrasting the Method of Characteristics which only works for specific cases the Winslow method [**6**] is a more general method. It targets problem areas as indicated by a monitor function $\omega : \Omega \to \mathbb{R}^+$. The higher the monitor function the higher the node density at that point.

6.1. One dimensional case.

$$x_\theta = \frac{1}{\tau}(\omega x_\xi)_\xi, \tag{6.1}$$

with

$$\omega(t, \mathbf{x}) := \alpha(t) + |u_\xi|, \tag{6.2}$$

and

$$\alpha(t) := \int_{x_{\min}}^{x_{\max}} |u_\xi| \,\mathrm{d}x. \tag{6.3}$$

Discretize using BTCS discretization:

$$(\sigma_{B\theta}\mathbf{x})^n = \frac{1}{\tau}\sigma_{(C\xi)^{\frac{1}{2}}}(\mathrm{diag}(\boldsymbol{\omega}^{n-1})\left(\sigma_{(C\xi)^{\frac{1}{2}}}\mathbf{x}\right)^n) \tag{6.4}$$

$$\begin{aligned}
\frac{\mathbf{x}_i^n - \mathbf{x}_i^{n-1}}{h_\theta} &= \frac{1}{\tau}\frac{(\mathrm{diag}(\boldsymbol{\omega}^{n-1})\sigma_{(C\xi)^{\frac{1}{2}}}\mathbf{x})_{i+1/2}^n - (\mathrm{diag}(\boldsymbol{\omega}^{n-1})\sigma_{(C\xi)^{\frac{1}{2}}}\mathbf{x})_{i-1/2}^n}{h_\xi} \\
\frac{\mathbf{x}_i^n - \mathbf{x}_i^{n-1}}{h_\theta} &= \frac{1}{\tau}\frac{\boldsymbol{\omega}_{i+1/2}^{n-1}\frac{\mathbf{x}_{i+1}^n - \mathbf{x}_i^n}{h_\xi} - \boldsymbol{\omega}_{i-1/2}^{n-1}\frac{\mathbf{x}_i^n - \mathbf{x}_{i-1}^n}{h_\xi}}{h_\xi} \\
\frac{\mathbf{x}_i^n - \mathbf{x}_i^{n-1}}{h_\theta} &= \frac{1}{\tau}\frac{\left(\alpha^{n-1} + \left|(\sigma_{C/2\xi}\mathbf{u})_{i+1/2}^{n-1}\right|\right)\frac{\mathbf{x}_{i+1}^n - \mathbf{x}_i^n}{h_\xi} - \left(\alpha^{n-1} + \left|(\sigma_{(C\xi)^{\frac{1}{2}}}\mathbf{u})_{i-1/2}^{n-1}\right|\right)\frac{\mathbf{x}_i^n - \mathbf{x}_{i-1}^n}{h_\xi}}{h_\xi} \\
\frac{\mathbf{x}_i^n - \mathbf{x}_i^{n-1}}{h_\theta} &= \frac{1}{\tau}\frac{\left(\alpha^{n-1} + \left|\mathbf{u}_{i+1}^{n-1} - \mathbf{u}_i^{n-1}\right|\right)\frac{\mathbf{x}_{i+1}^n - \mathbf{x}_i^n}{h_\xi} - \left(\alpha^{n-1} + \left|\mathbf{u}_i^{n-1} - \mathbf{u}_{i-1}^{n-1}\right|\right)\frac{\mathbf{x}_i^n - \mathbf{x}_{i-1}^n}{h_\xi}}{h_\xi}.
\end{aligned}$$

Note that applying the $\sigma_{(C\xi)^{\frac{1}{2}}}$ operator twice results in the appearance of the $\sigma_{F\xi}$ and $\sigma_{B\xi}$ operators:

$$\frac{\mathbf{x}_i^n - \mathbf{x}_i^{n-1}}{h_\theta} = \frac{1}{\tau}\frac{\left(\alpha^{n-1} + \left|(\sigma_{F\xi}\mathbf{u})_i^{n-1}\right|\right)\frac{\mathbf{x}_{i+1}^n - \mathbf{x}_i^n}{h_\xi} - \left(\alpha^{n-1} + \left|(\sigma_{B\xi}\mathbf{u})_i^{n-1}\right|\right)\frac{\mathbf{x}_i^n - \mathbf{x}_{i-1}^n}{h_\xi}}{h_\xi},$$

which will be stored in $\boldsymbol{\omega}_F$ and $\boldsymbol{\omega}_B$.

$$\begin{aligned}(\boldsymbol{\omega}_F)_i^n &:= \alpha^{n-1} + \left|(\sigma_{F\xi}\mathbf{u})_i^{n-1}\right|,\\ (\boldsymbol{\omega}_B)_i^n &:= \alpha^{n-1} + \left|(\sigma_{B\xi}\mathbf{u})_i^{n-1}\right|.\end{aligned}$$

The discretization for the one dimensional Winslow equation can now be formulated as:

$$\frac{\mathbf{x}_i^n - \mathbf{x}_i^{n-1}}{h_\theta} = \frac{1}{\tau}\frac{(\boldsymbol{\omega}_F)_i^{n-1}\left(\mathbf{x}_{i+1}^n - \mathbf{x}_i^n\right) - (\boldsymbol{\omega}_B)_i^n\left(\mathbf{x}_i^n - \mathbf{x}_{i-1}^n\right)}{h_\xi^2}.$$

6.2. ***d* dimensional case.** In d dimensional case d equations are formulated for all d coordinates.

$$\left\{\begin{aligned}(x_1)_\theta &= \tfrac{1}{\tau}\nabla_{\boldsymbol{\xi}}\cdot(\omega(u)\nabla_{\boldsymbol{\xi}}x_1),\\ &\vdots\\ (x_d)_\theta &= \tfrac{1}{\tau}\nabla_{\boldsymbol{\xi}}\cdot(\omega(u)\nabla_{\boldsymbol{\xi}}x_d)\end{aligned}\right.,$$

with monitor function

$$\omega(u) = \alpha(u) + \beta_\gamma(u), \tag{6.5}$$

and

$$\beta_\gamma(u) := ||\nabla_{\boldsymbol{\xi}}u||^{\frac{1}{\gamma}} \tag{6.6}$$

$$= \left(\sum_{k=1}^{d}\left(\frac{\partial u}{\partial \xi_k}\right)^2\right)^{\frac{1}{2\gamma}} \tag{6.7}$$

and

$$\alpha(u) := \int_\Omega \beta_\gamma(u)\,\mathrm{d}\mathbf{x} \tag{6.8}$$

$$= \int_{(x_{\min})_1}^{(x_{\max})_1}\dots\int_{(x_{\min})_d}^{(x_{\max})_d}\beta_\gamma(u)\,\mathrm{d}x_d\dots\mathrm{d}x_1. \tag{6.9}$$

Discretization yields:

$$(\sigma_{B\theta}\mathbf{x}_k)^n = \frac{1}{\tau}\sum_{k'=1}^{d}\sigma_{(C\xi_{k'})^{\frac{1}{2}}}\left(\mathrm{diag}(\boldsymbol{\omega}^{n-1})\left(\sigma_{(C\xi_{k'})^{\frac{1}{2}}}\mathbf{x}_k\right)^n\right) \tag{6.10}$$

$$\Rightarrow \quad \frac{(\mathbf{x}_k)^n_i - (\mathbf{x}_k)^{n-1}_i}{h_\theta} =$$
$$\frac{1}{\tau}\sum_{k'=1}^{d} \frac{(\boldsymbol{\omega})^{n-1}_{i-\Delta\frac{1}{2}\mathbf{e}_{k'}}\left((\mathbf{x}_k)^n_{i+\Delta\mathbf{e}_{k'}} - (\mathbf{x}_k)^n_i\right) - (\boldsymbol{\omega})^{n-1}_{i+\Delta\frac{1}{2}\mathbf{e}_{k'}}\left((\mathbf{x}_k)^n_i - (\mathbf{x}_k)^n_{i-\Delta\mathbf{e}_{k'}}\right)}{h_\xi^2}.$$

In this equation we recognize terms from the one dimensional case. But a complication arises as $\boldsymbol{\omega}$ is evaluated not only in $i \pm 1/2$ but in $i \pm \Delta\frac{1}{2}\mathbf{e}_{k'}$. In the once dimensional case this was compensated by discretizing the u_ξ term in ω with $\sigma_{(C\xi)^{\frac{1}{2}}}$ resulting in only integer node values. For the $k = k'$ terms the addition or subtraction of two $\frac{1}{2}$ terms will result in integer node values but for $k \neq k'$ non integer (in between) node values will remain. As a solution I implement a linear interpolation of the surrounding integer nodes along the $x_{k'}$-axis:

$$\boldsymbol{\omega}^{n-1}_{i\pm\Delta\frac{1}{2}\mathbf{e}_{k'}} = \left(\alpha^{n-1} + \left|\sigma_{(C\xi_{k'})^{\frac{1}{2}}}\mathbf{u}\right| + \frac{1}{2}\sum_{\substack{k''=1\\k''\neq k'}}^{d}\left|(Q^{\frac{1}{2}\mathbf{e}_{k'}}\sigma_{Cx_{k''}} + Q^{-\frac{1}{2}\mathbf{e}_{k'}}\sigma_{Cx_{k''}})\mathbf{u}\right|\right)^{n-1}_{i\pm\Delta\frac{1}{2}\mathbf{e}_{k'}}.$$

These results will be stored in vectors $\boldsymbol{\omega}_{Bx_{k'}}$ and $\boldsymbol{\omega}_{Fx_{k'}}$ for $1 \leq k' \leq d$.

$$\begin{aligned}
\left(\boldsymbol{\omega}_{F\xi_{k'}}\right)^n_i &:= \alpha^{n-1} + \left|(\sigma_{F\xi_{k'}}\mathbf{u})^{n-1}_i\right| + \frac{1}{2}\sum_{\substack{k''=1\\k''\neq k'}}^{d}\left|(\sigma_{C\xi_{k''}}\mathbf{u})^{n-1}_{i+\Delta\mathbf{e}_{k'}} + (\sigma_{C\xi_{k''}}\mathbf{u})^{n-1}_i\right| \\
\left(\boldsymbol{\omega}_{B\xi_{k'}}\right)^n_i &:= \alpha^{n-1} + \left|(\sigma_{F\xi_{k'}}\mathbf{u})^{n-1}_i\right| + \frac{1}{2}\sum_{\substack{k''=1\\k''\neq k'}}^{d}\left|(\sigma_{C\xi_{k''}}\mathbf{u})^{n-1}_i + (\sigma_{C\xi_{k''}}\mathbf{u})^{n-1}_{i-\Delta\mathbf{e}_{k'}}\right|
\end{aligned}$$

The discretization for the multi-dimensional Winslow equation can now be formulated as:

$$\frac{(\mathbf{x}_k)^n_i - (\mathbf{x}_k)^{n-1}_i}{h_\theta} =$$
$$\frac{1}{\tau}\sum_{k'=1}^{d} \frac{(\boldsymbol{\omega}_{F\xi_{k'}})^{n-1}_i\left((\mathbf{x}_k)^n_{i+\Delta\mathbf{e}_{k'}} - (\mathbf{x}_k)^n_i\right) - (\boldsymbol{\omega}_{B\xi_{k'}})^{n-1}_i\left((\mathbf{x}_k)^n_i - (\mathbf{x}_k)^n_{i-\Delta\mathbf{e}_{k'}}\right)}{h_{\xi_{k'}}^2}.$$

6.3. Considerations on dimensional comparison. The higher the dimension the larger the size of the d-dimensional sphere around a point and the more 'heat' can be transported from that point. The equalizing effect of diffusion will be 'stronger' the higher the dimension of the domain. Since the problem will already be harder (computation and memory wise) as the dimension increases (since more nodes are needed to maintain the same node density) it seems unfair, or even unreasonable, to also maintain a constant diffusion coefficient on top of what is called *the Curse of dimensionality* . The fact that a problem becomes harder when the dimension of the domain or the diffusion coefficient is increased remains but to create a comparison I decided to implement a dimensionally equalizing diffusion coefficient $\kappa = d^{-1}$. This choice for κ results in a stability criterium independent of d and an exact solution for the d-dimensional case that is, when restricted to a lower dimensional domain, equal to the lower dimensional solution over that domain.

Contrary to diffusion advection does not increase with the dimension of the domain. A directional flow does not increase when placed in a higher dimensional volume. No coefficients are needed to ensure that the exact solution for the d-dimensional case is, when restricted to a lower dimensional domain, equal to the lower dimensional solution over that domain.

6.4. Considerations on method comparison. To show the relevance of the Winslow versus the uniform approximation of the three test cases (cf. Appendix 3) I endeavored to construct a comparison as follows:

(1) Choose a time step resolution h_θ and a base node count m for the Winslow method.

(2) Determine the memory load, computation time and maximum error for a given dimension, d.

(3) Find base node count m_1 for the uniform method such that the memory load roughly equals that of the Winslow method.

(4) Find a time step resolution h_t for the uniform method such that the computation time roughly equals that of the Winslow method.

(5) Compare the maximum error.

The method that results in smallest maximum error can be argued to be the more efficient method: with roughly the same resources (time and memory) it achieves a higher accuracy.

The uniform base node count, m, in step (3) is the number of nodes in each axial direction making the total number of nodes m^d. It is derived analytically while the uniform time resolution, h_t, in step (4) is found empirically.

For step (3) estimates for the memory use are formulated for both methods. The memory needed for the uniform implementation is described by: $6m^d$. The factor 6 comes from the storing the solution and 5 vectors needed for the BiCGStab iteration. For the Winslow method the memory requirements are described by: $(6+2d+2d(d+1))m^d$ The factor 6 covers the same vectors as the uniform case but is supplemented with $2d$ vectors for storing the d-dimensional node locations at step n and $n-1$. The $2d(d+1)$ vectors are used to store the Jacobi matrix again at n and $n-1$. Equating these formulas results and solving for adaptive axial node count of 19 (19 nodes in each direction, 19^d nodes in total) yields: $38, 36.3822, 34.5253, 32.909, 31.5708$ for $d = 1, 2, 3, 4, 5$ Rounding up to the highest odd integer yields a uniform axial node count of 39.

The time stepsize for the Winslow method $h_\theta = \frac{T}{100}$ such that it takes 100 steps to iterate to end time T.

6.5. Results. To asses and compare the performance of the uniform and adaptive methods numerical experiments for the three test cases were performed. Al experiments were run on a 2008 MacBook 2.4 Ghz Intel Core 2 Duo, although no parallelization has yet been implemented in the software to benefit from the dual core. The time measurements represent the CPU time used by the program.

The error estimates are made using the exact solutions. A maximum of this error is determined over a one dimensional cross section of the domain: $\xi_2 = \dots \xi_d = 0$ for which the solutions in each test case is equal throughout the dimension as stated in .

6.5.1. *Testcase 1 : Diffusion.* To roughly match and at least surpass the computation time for the Winslow method a time step size for the uniform method is chosen as $h_\theta = \frac{T}{2500}$.

d	Computation time (ms)		max. error	
	Winslow	Uniform	Winslow	Uniform
1	2.5×10^1	5.8×10^1	0.010	0.0021
2	3.2×10^2	5.0×10^2	0.0084	0.0021
3	9.9×10^3	1.2×10^4	0.0075	0.0021
4	5.0×10^5	9.6×10^5	0.0071	0.0021
5	3.7×10^7	6.2×10^7	0.0070	0.0021

TABLE 1. Results for Testcase 1 : Heat

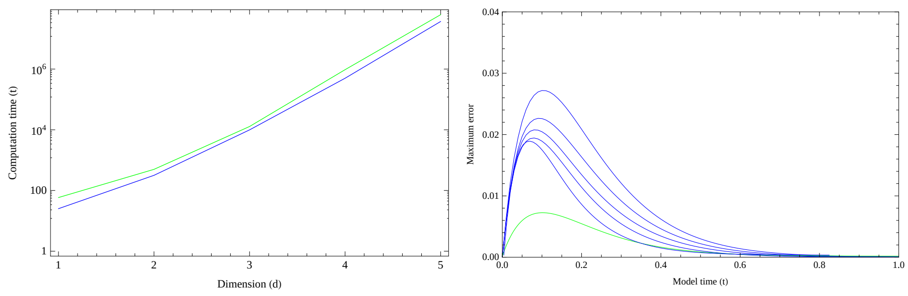


FIGURE 1. Method comparison of the d-dimensional Heat equation (Testcase 1). Uniform (Green) 39^d and Winslow (Blue) 19^d nodes.

This clearly shows that the adaptive approach underperforms compared to the uniform method. The marginal adaption visible in Figure 2 compared to the uniform method indicates that the transformation yields little difference but still requires the increased computational load of the Winslow method. This is caused by the little variation in the first order derivative of the solution. On top of that the extra terms of the transformation of the second order derivatives as mentioned in section 3.3 are cause for increased computation time and further reduce the efficiency.

6.5.2. *Testcase 2 : Implosion.* To roughly match and at least surpass the computation time for the Winslow method a time step size for the uniform method is chosen as $h_\theta = \frac{T}{1000}$.

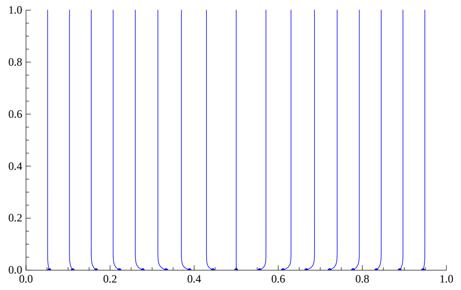


FIGURE 2. The inner nodes $\mathbf{q}(\urcorner)_1$ such that $\xi_2 = \ldots \xi_d = 0$ as a function of time θ. A one dimensional cross section of the $d = 5$ case.

d	Computation time (ms)		max. error	
	Winslow	Uniform	Winslow	Uniform
1	5.3×10^1	7.0×10^2	0.25	0.92
2	2.5×10^2	3.5×10^2	0.21	0.92
3	9.5×10^3	1.4×10^4	0.23	0.92
4	3.9×10^5	7.8×10^5	0.23	0.92
5	2.2×10^7	9.3×10^7	0.22	0.92

TABLE 2. Results for Testcase 2 : Implosion

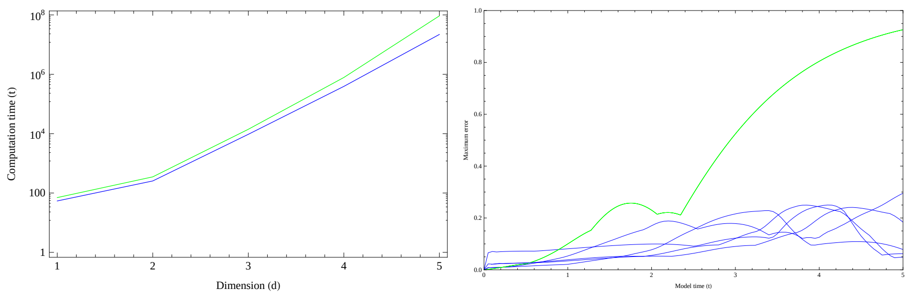


FIGURE 3. Method comparison of the d-dimensional Implosion equation (Testcase 2). Uniform (Green) 39^d and Winslow (Blue) 19^d nodes.

Due to the steepness the Implosion test case the Winslow method performs considerably better. The method of Characteristics performs even better, in fact so good that its error cannot be significantly depicted in the graph, but this is to be expected of a method that actually prescribes an analytical solution. The node lines in Figure 4 clearly indicate that the system is strongly adapted resulting in increased overall performance.

6.5.3. *Testcase 3 : Burgers'.* The steepness or non-smoothness of the Burgers' solution is between that of the two previous test cases. The inclusion of second order derivative increases the computational workload as it did with the Heat test case. The node lines in Figure 6 clearly indicate the traveling wave and show adaptivity. The benefits from this adaptivity is enough to outperform the uniform method based on the maximum error. On top of that the uniform solution shows small irregularities

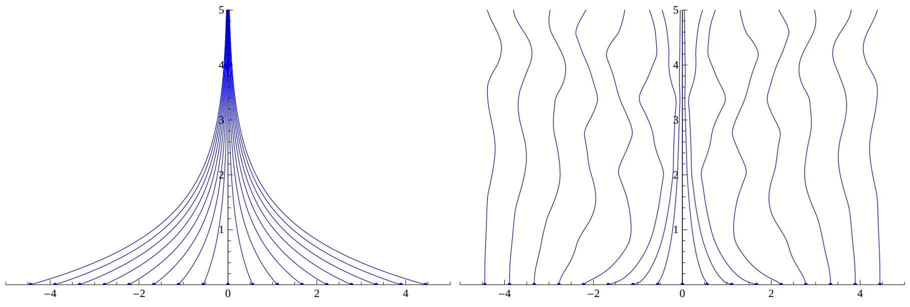


FIGURE 4. The inner nodes $\mathbf{q}(\lrcorner)_1$ such that $\xi_2 = \ldots \xi_d = 0$ as a function of time θ. Method of Characteristics left, Winslow Method right.

d	Computation time (ms)		max. error	
	Winslow	Uniform	Winslow	Uniform
1	5.5×10^1	6.4×10^2	0.30	0.44
2	5.1×10^2	4.4×10^2	0.24	0.44
3	1.3×10^4	1.4×10^4	0.22	0.44
4	5.6×10^5	1.0×10^6	0.21	0.44
5	5.4×10^7	8.1×10^7	0.16	0.44

TABLE 3. Results for Testcase 3 : Burgers'

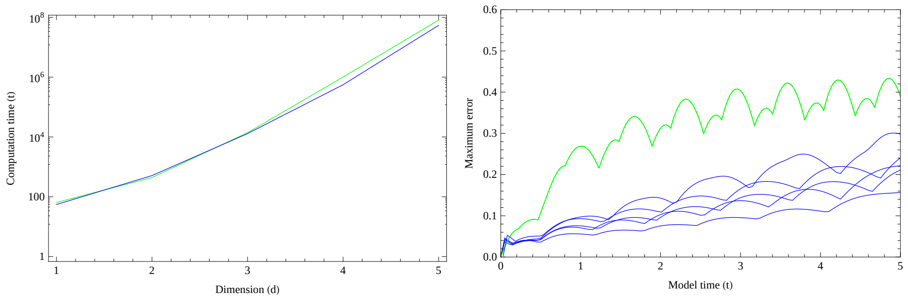


FIGURE 5. Method comparison of the d-dimensional Burgers' equation (Testcase 3). Uniform (Green) 39^d and Winslow (Blue) 19^d nodes.

or 'wiggles' on top of the solution which are undesirable. The error in the Winslow method seems to be mainly based on a numerical solution that 'drags' behind the exact solution.

6.6. Conclusion. The use of general definitions for the uniform domain discretization facilitated easy implementation of higher dimensional r-refinement methods. By using the same methods for the grid and the transformed equation the generality and flexibility of the software can be fully utilized. Providing options for higher order, non linear and non zero, non static boundaries.

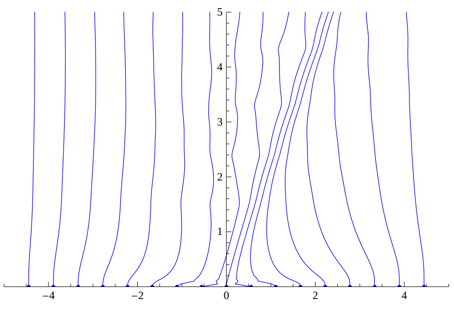


FIGURE 6. The inner nodes $\mathbf{q}(\natural)_1$ such that $\xi_2 = \ldots \xi_d = 0$ as a function of time θ.

The numerical results from the test cases show that the adaptive method can deliver a higher quality approximation at the same or even lower cost than the uniform approach. Validating the implementation of the more complex computations for the adaptive method. At this point no exact formulations can be made to predict wether a method will benefit from r-refinement. The adaptive method seems to outperform the uniform method depending on the turbulence. The more pronounced differences in the first order derivative are the better method can adapt to problematic areas.

Especially higher dimensional problems can benefit from adaptive approaches since any reduction in required node density will pay of relatively exponentially. Higher order higher dimensional derivatives do pose a problem since their transformation yields an exponential increasing number of approximation terms. The diffusion terms for higher dimensional problems can also proof to be more difficult to approximate when they are not scaled with a d^{-1} parameter as was done for the test cases.

Using the limited computing capabilities of a personal computer a five dimensional application remains the upper limit. But even with increased computation power it seems unavoidable that further improvements should be made to allow constructive methods for even higher dimensions or higher accuracy. Hopefully the generalized definitions can facilitate in the development of these methods.

6.7. Future Research. Since higher dimensional problems remain a heavy task further improvements remain very desirable. Implementation of the simultaneous solving of several traversal layers would yield possibilities for Neumann boundaries, p-refinement, improved r-refinement by simultaneous solving of v and $\mathbf{q}$ and even vector valued solutions.

Simultaneous parallel solving would be a welcome extension to distribute the workload amongst multiple processor units. Definitions for matrix representations can be used to construct communication matrices.

Exploring the matrix structures arising from h-refinement or skewed uniform grids for anisotropic applications and thus venturing into non uniform grids for which the notation already provides, might yield interesting adaptive extensions. r-refinement can benefit greatly from improved monitor functions.

A dimension independent monitor function that provides similar results throughout dimensions would fine tuning considerably easier. Using more terms of the computed truncation function as a basis for the monitor function might provide a more accurate handling of problem areas.

Bibliography

[1] C. Reisinger and G. Wittum, Efficient hierarchical approximation of high-dimensional option pricing problems, *SIAM Journal on Scientific Computing* **29(1)** (2007), 440-458.

[2] P. Sjöberg, P. Lötstedt, and J. Elf, Fokker-Planck approximation of the master equation in molecular biology, Technical Report **2005-044** (2005) Uppsala Universitet.

[3] C. Leforestier, R.H. Bisseling, C. Cerjan, M.D. Feit, R. Friesner, A. Guldberg, A. Hammerich, G. Jolicard, W. Karrlein, H.-D. Meyer, N. Lipkin, O.Roncero, and R. Kosloff, A comparision of different propagation schemes for the time dependent Schrödinger equation, *Journal of Computational Physics* **94** (1991), 59-80.

[4] Van der Vorst, H. A., Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems, *SIAM J. Sci. and Stat. Comput.* **13 (2)** (1992) 631644

[5] Evans, Lawrence C., Partial Differential Equations, *American Mathematical Society* (1998) ISBN 0-8218-0772-2

[6] A. Winslow, Numerical solution of the quasi-linear Poisson equation, *Journal of Computational Physics* (1967) 149-172.

[7] P.A. Zegeling, Theory and Application of Adaptive Moving Grid Methods, *Adaptive Computations: Theory and Algorithms*, edited by T. Tang and J. Xu (2007)

[8] A. van Dam, and P.A. Zegeling, Balanced Monitoring of Flow Phenomena in Moving Mesh Methods, *Comput. Phys.* **7** (2010), 138-170

[9] The eigenvalues of tridiagonal matrices. `http://www.cems.uvm.edu/~tlakoba/math337/proof_eigensystem_tridiagonal.pdf` *Note TMA4205*, (2009) Norwegian University of Science and Technology

[10] Randall J. LeVeque Finite Difference Methods for Differential Equations, *AMath 585-6, University of Washington, Winter/Spring Quarters*, (1998)

CHAPTER 5

Appendix

1. Stencil Operations

A stencil can be viewed as a sparse notation for both differential and finite difference operators. But also for the matrices that arise in the finite difference methods. Sthe stencil operations and notation is defined similarly for the differential and finite difference operators. The only differences arise for local scaling: a differential operator is multiplied with a function returning individual values for each $\mathbf{q}$ in domain Ω; a finite difference operator is multiplied by the discretization of a function returning individual values for each node index $i \in \mathcal{I}$.

A stencil is as set of tuples containing a scalar, s, and a coefficient, j:

label coefficient

$$S = \bigcup_i (s_i, \mathbf{j}_i) \tag{1.1}$$

Addition is equivalent to a union:

$$S_1 + S_2 \quad := \quad S_1 \cup S_2 \tag{1.2}$$

Global scalar multiplication with a scalar α can easily be defined as:

$$\alpha \bigcup_k \{(s_k, j_k)\} := \bigcup_k \{(\alpha s_k, j_k)\} \tag{1.3}$$

Subtraction is performed by combining a union and scalar multiplication by -1.

Local (per node) scalar multiplication with a (discretized) function $\alpha : \mathcal{A} \to \mathbb{R}$ is for differential operators is noted as:

$$\alpha(\cdot) \bigcup_k \{(s_k, j_k)\}\mathbf{f}^{\mathcal{A}} = \bigcup_k \{(\alpha(\cdot)s_k, j_k)\}\mathbf{f}^{\mathcal{A}} \tag{1.4}$$

and for finite difference operators as:

$$\operatorname{diag}(\boldsymbol{\alpha}^{\mathcal{A}}) \bigcup_k \{(s_k, j_k)\}\mathbf{f}^{\mathcal{A}} = \bigcup_k \{(\alpha(\cdot)s_k, j_k)\}\mathbf{f}^{\mathcal{A}} \tag{1.5}$$

Multiplication or more accurately *composition* of two operators is defined separately for general finite difference operators

$$\left(\bigcup_k \{(s_k, j_k)\}\right) \circ \left(\bigcup_{k'} \{(s'_{k'}, j'_{k'})\}\right) := \bigcup_k \bigcup_{k'} \{(s_k s'_{k'}, j_k \Delta_{i+\Delta_i j_k} j'_{k'})\} \tag{1.6}$$

The new scalars will be formed by the multiplication of all combinations of the scalars. The new coefficients are best viewed as the adjacency label relative from node i of the node that is the j' neighbor of the j neighbor of i.

When a discretization is *transitive* this will adhere to the much simpler form:

$$\left(\bigcup_k \{(s_k, j_k)\}\right) \circ \left(\bigcup_{k'} \{(s'_{k'}, j'_{k'})\}\right) := \bigcup_k \bigcup_{k'} \{(s_k s'_{k'}, j_k + j'_{k'})\} \tag{1.7}$$

The latter form is also applicable for differential operators.

To speed up calculations every two terms of S for which it holds that $j_{k_1} = j_{k_2}$ can be combined into a single term $(s_{k_1} + s_{k_2}, j_{k_1})$.

Inverse scaling can only be computed trivially for stencils of the form $S = (s, \mathbf{0})$ with $s \neq 0$. $\frac{1}{S} = (\frac{1}{s}, \mathbf{0})$ for scalars, $\frac{1}{S} = (\frac{1}{s(\cdot)}, \mathbf{0})$ for functions which yields the inverse of a diagonal, $\operatorname{diag}(\mathbf{s})^{-1}$ again a diagonal matrix for discretized functions.

Inverse composition should yield (approximations of) anti derivatives or primitive integrals.

$$\int^b a \frac{\partial}{\partial x} f \, \mathrm{d}x \quad = \quad f(b) - f(a) \tag{1.8}$$

Or alternatively:

$$\int_a^b f \, \mathrm{d}x \quad = \quad \left(\frac{\partial}{\partial x}\right)^{-1} f(b) - \left(\frac{\partial}{\partial x}\right)^{-1} f(a) \tag{1.9}$$

For a one dimensional uniform domain discretizaions it follows that

$$\int_{x_i}^{x_j} f \, \mathrm{d}x \quad \approx \quad (M_{Fx})^{-1} \mathbf{f}_i - (M_{Fx})^{-1} \mathbf{f}_j \tag{1.10}$$

But further research is needed to explore extensions to general domain dicretizations and higher dimensions.

2. Matrices and vectors

Definition 17. *The* ***stack*** *and* ***append*** *operators are used to construct matrices. Horizontal and vertical stacking are defined by:*

$$a \boxed{\rightarrow} b \quad = \quad \begin{pmatrix} a & b \end{pmatrix} \tag{2.1}$$

$$a \boxed{\downarrow} b \quad = \quad \begin{pmatrix} a \\ b \end{pmatrix} \tag{2.2}$$

while horizontal and vertical appending is defined by:

$$(2.3)\qquad \begin{pmatrix} a & b \end{pmatrix} \boxed{\Rightarrow} c \;=\; \begin{pmatrix} a & b & c \end{pmatrix}$$

$$(2.4)\qquad \begin{pmatrix} a \\ b \end{pmatrix} \boxed{\Downarrow}\; c \;=\; \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Note that the left hand side of the append operators should be a $n \times m$-matrix (or vector) and the right hand side should be compatibly shaped: $p \times m$-matrix for horizontal appending and a $n \times q$-matrix for vertical appending.

When applied to matrices $A := \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $B := \begin{pmatrix} e & f \\ g & h \end{pmatrix}$ the differences become more apparent:

$$(2.5)\qquad A \boxed{\Rightarrow} B \;=\; \begin{pmatrix} a & b & e & f \\ c & d & g & h \end{pmatrix}$$

$$(2.6)\qquad A \boxed{\rightarrow} B \;=\; \begin{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} & \begin{pmatrix} e & f \\ g & h \end{pmatrix} \end{pmatrix}$$

The first is a 2×4-matrix of scalars the second a 2×1 vector of 2×2 matrices. They are very different entities as demonstrated by:

$$(2.7)\qquad \begin{pmatrix} a & b & e & f \\ c & d & g & h \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \;=\; \begin{pmatrix} xa + yb + ze + df \\ xc + yd + zg + wh \end{pmatrix}$$

$$(2.8)\qquad \begin{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} & \begin{pmatrix} e & f \\ g & h \end{pmatrix} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \;=\; \begin{pmatrix} a & b \\ c & d \end{pmatrix} x + \begin{pmatrix} e & f \\ g & h \end{pmatrix} y$$

$$=\; \begin{pmatrix} xa & xb & ey & fy \\ cx & dx & gy & hy \end{pmatrix}$$

The operators can also be used in repeated form. Let $A_1, \ldots, A_n$ be matrices. If they are compatibly shaped a block matrix can be formed by using the repeated append operator:

$$(2.9)\qquad \overset{n}{\underset{i=1}{\boxed{\Rightarrow}}} A_i \;=\; A_1 \boxed{\Rightarrow} \ldots \boxed{\Rightarrow} A_n = (A_1 \ldots A_n)$$

$$(2.10)\qquad \overset{n}{\underset{i=1}{\boxed{\Downarrow}}}\; A_i \;=\; A_1 \boxed{\Downarrow}\; \ldots \boxed{\Downarrow}\; A_n = \begin{pmatrix} A_1 \\ \vdots \\ A_n \end{pmatrix}$$

Let $a_1, \ldots, a_n$ be scalars then repeating the stack operator yields the following nested construction:

$$(2.11)\qquad a_1 \boxed{\rightarrow} a_2 \boxed{\rightarrow} a_3 = \begin{pmatrix} a_1 & a_2 \end{pmatrix} \boxed{\rightarrow} a_3 = \begin{pmatrix} \begin{pmatrix} a_1 & a_2 \end{pmatrix} & a_3 \end{pmatrix}$$

Because I have had no need for such a nested constructing I define the repeated stack operator as:

$$\overset{n}{\underset{i=1}{\boxed{\rightarrow}}} a_i \;=\; a_1 \boxed{\rightarrow} a_2 \boxed{\Rightarrow} a_3 \boxed{\Rightarrow} \ldots \boxed{\Rightarrow} a_n = (a_1 \ldots a_n) \tag{2.12}$$

$$\overset{n}{\underset{i=1}{\boxed{\downarrow}}}\; a_i \;=\; a_1 \boxed{\downarrow}\; a_2 \boxed{\Downarrow}\; a_3 \boxed{\Downarrow}\; \ldots \boxed{\Downarrow}\; a_n = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \tag{2.13}$$

where the repeated appending operator can only be applied to matrices, the repeated stacking operator can also be applied to scalars. Note that when the repeated stacking operator is applied to matrices it will result in a matrix of matrices.

A d-dimensional vector $\mathbf{v}$ *with values* v_k *can thus be denoted as:*

$$\mathbf{v} \;=\; \overset{d}{\underset{k=1}{\boxed{\downarrow}}}\; v_k \tag{2.14}$$

$$\mathbf{v}^T \;=\; \overset{d}{\underset{k=1}{\boxed{\rightarrow}}} v_k \tag{2.15}$$

and a $m \times n$ *matrix* A *with values* $a_{i,j}$ *can be be denoted as:*

$$A \;=\; \overset{n}{\underset{i=1}{\boxed{\downarrow}}}\; \overset{m}{\underset{j=1}{\boxed{\rightarrow}}} a_{i,j} \tag{2.16}$$

$$A^T \;=\; \overset{n}{\underset{i=1}{\boxed{\downarrow}}}\; \overset{m}{\underset{j=1}{\boxed{\rightarrow}}} a_{j,i} \tag{2.17}$$

$$\tag{2.18}$$

Definition 18. *The* ***Kronecker delta****,* δ*, is defined an function that returns one or zero based on the equality of its two subscript parameters:*

$$\delta_{a,b} = \begin{cases} 1 & a = b \\ 0 & \textit{otherwise} \end{cases}$$

Definition 19. *The* ***base vectors****,* $\mathbf{e}_k$*, are d-dimensional vectors. Their k-th element is one, all others are zero. d is defined by context.*

$$(\mathbf{e}_k)_{k'} = \begin{cases} 1 & k = k' \\ 0 & \textit{otherwise} \end{cases}$$

Or using stack notation and the previously defined Kronecker delta*:*

$$\mathbf{e}_k \;=\; \overset{d}{\underset{k'=1}{\boxed{\downarrow}}}\; \delta_{k,k'} \tag{2.19}$$

Definition 20. *The **identity matrix**, I_m, is $m \times m$-matrix with ones along its diagonal and zeroes elsewhere. Subscript m will be left out if it can be determined from context.*

$$(I_m)_{i,j} = \begin{cases} 1 & j = i \\ 0 & \textit{otherwise} \end{cases} \tag{2.20}$$

Or using stack notation:

$$I_m \quad = \quad \underset{i=1}{\overset{m}{\boxed{\downarrow}}}\ \underset{j=1}{\overset{m}{\boxed{\rightarrow}}}\delta_{i,j} \tag{2.21}$$

Definition 21. *The **zero matrix**, $\mathbf{0}_{m_1,m_2}$, and **one matrix**, $\mathbf{1}_{m_1,m_2}$, are $m_1 \times m_2$-matrix consisting of all zeroes respectively ones. Subscripts m_1 and m_2 will be left out if it can be determined from context. When only m_2 is left out it is considered equal to one indicating the **zero vector**, respectively **one vector**.*

$$(\mathbf{0}_{m_1,m_2})_{i,j} = 0 \tag{2.22}$$

Or using stack notation:

$$\mathbf{0}_{m_1,m_2} \quad = \quad \underset{i=1}{\overset{m_1}{\boxed{\downarrow}}}\ \underset{j=1}{\overset{m_2}{\boxed{\rightarrow}}}0 \tag{2.23}$$

Definition 22. *The **offset matrix**, J_m, is $m \times m$-matrix with ones on the above diagonal and zeroes elsewhere. Subscript m will be left out if it can be determined from context.*

$$(J_m)_{i,j} = \begin{cases} 1 & j = i + 1 \\ 0 & \textit{otherwise} \end{cases}$$

Or using stack notation:

$$J_m \quad = \quad \underset{i=1}{\overset{m}{\boxed{\downarrow}}}\ \underset{j=1}{\overset{m}{\boxed{\rightarrow}}}\delta_{i,j+1} \tag{2.24}$$

Example 10.

$$J_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Definition 23. *J_m^k is a special notational convention/abuse for the $m \times m$ k offset-matrix. It is defined as:*

$$J_m^k = \begin{cases} J_m^k & k > 0 \\ I_m & k = 0 \\ (J_m^T)^{-k} & k < 0 \end{cases}$$

Note that J_m^{-1} might be confused with the inverse of J_m

Example 11.

$$J_4^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad J_4^3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad J_4^{-2} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Lemma 5. $(J_m^k)_{i,j} = \begin{cases} 1 & j = i + k \\ 0 & \textit{otherwise} \end{cases}$

Corrolary 1. *Row i of matrix J_m^k has a non zero value if and only $i \in \{-k, \ldots, m-k\}$ and $i \in \{1, \ldots, m\}$*

Lemma 6. $aJ_m + bJ_m^T = \frac{a+b}{2}(J_m + J_m^T) + \frac{a-b}{2}(J_m - J_m^T)$

Corrolary 2. *A vector multiplication with J^j can be computed as a simple index shift. This allows for fast computations which will be used in further algorithms*

$$J^j \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} x_{1+j} \\ x_{2+j} \\ \vdots \\ x_{m+j} \end{pmatrix} \tag{2.25}$$

where $x_i = 0$ for $i \notin \{1, \ldots, m\}$

Definition 24. *The **boundary offset matrix** ∂J_m is a $m \times 1$-matrix which is structured as follows:*

$$\partial J_m := \mathbf{e}_1 \tag{2.26}$$

Example 12.

$$\partial J_4 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Definition 25. *∂J_m^k is a special notational convention/abuse for the $m \times m$ or $2 \times m$ k boundary offset-matrix. It is defined as:*

$$\partial J_m^k := \begin{cases} I_m & k = 0 \\ \mathbf{e}_{m-k} & k > 0 \\ \mathbf{e}_{-k} & k < 0 \end{cases} \tag{2.27}$$

Note that ∂J_m^{-1} might be confused with a (pseudo) inverse of ∂J_m

Example 13.

$$\partial J_4^2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \qquad \partial J_4^3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \partial J_4^{-2}, \qquad \partial J_4^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Eigenvalues and Matrix Norms

DEFINITION 26. *The **spectrum** of a $m \times m$-matrix A is denoted as $\lambda(A)$. It is the set consisting of all eigenvalues λ_k of A.*

Properties of spectra include:

- $\lambda(I) = \{1\}$
- $\lambda(sA) = s\lambda(A)$
- $\lambda(sI + A) = s + \lambda(A)$
- *For A a diagonal, upper of lower triangular $m \times m$-matrix:* $\lambda(A) = \bigcup_{i=1}^{m} A_{i,i}$
- $\lambda(A^k) = \{\lambda_i^k \mid \lambda_i \in \lambda(A)\}$
- $\lambda(A^{-1}) = \{\lambda_i^{-1} \mid \lambda_i \in \lambda(A)\}$

LEMMA 7. $\lambda(aJ_m + bJ_m^T) = \{2\sqrt{ab}\cos(\frac{\pi s}{m+1}) \mid s \in \{1, \ldots, m\}\} \subseteq 2\sqrt{ab}(-1, 1)$

[**9**]

DEFINITION 27. *The **matrix norm** is defined as the maximum length of the matrix vector product with all unit vectors:*

$$||A|| := \max_{||x||=1} ||Ax|| \tag{3.1}$$

Properties of the Matrix Norm include:

- $||A|| \geq 0$
- $||A|| = 0 \Leftrightarrow A = \mathbf{0}$
- $||A + B|| \leq ||A|| + ||B||$
- $||AB|| \leq ||A|| \, ||B||$

LEMMA 8. $\lambda(A) \subseteq \|A\| \, \mathbb{C}_{<1}$

where $\mathbb{C}_{<1}$ is the open unit disk in the complex plane.

Proof *This follows from the definition of eigenvalues. For each eigenvalue λ_i and eigenvector $\mathbf{v}_i$ of A it follows that*

$$\begin{aligned}
Av &= \lambda_i v && (3.2)\\
\Leftrightarrow A\frac{\mathbf{v}_i}{||\mathbf{v}_i||} &= \lambda_i \frac{\mathbf{v}_i}{||\mathbf{v}_i||} && (3.3)\\
\Rightarrow \exists \mathbf{x} \mid ||\mathbf{x}|| = 1 \textit{ such that } A\mathbf{x} &= \lambda_i \mathbf{x} && (3.4)\\
\Leftrightarrow \exists \mathbf{x} \mid ||\mathbf{x}|| = 1 \textit{ such that } ||A\mathbf{x}|| &= |\lambda_i| && (3.5)\\
\Rightarrow \max_{||vectx||=1} ||A\mathbf{x}|| &\geq |\lambda_i| && (3.6)\\
\Leftrightarrow ||A|| &\geq |\lambda_i| && (3.7)\\
\Leftrightarrow \lambda(A) \subseteq \|A\| \, \mathbb{C}_{<1} \quad \square & && (3.8)
\end{aligned}$$

LEMMA 9. *Matrix norm of the offset matrix*

$$||J_m^k|| = \begin{cases} 1 & |k| < m \\ 0 & \textit{otherwise} \end{cases}$$

COROLARY 3. $||J_m + J_m^T|| \leq 2$, $||J_m - J_m^T|| \leq 2$

LEMMA 10. *Eigenvalue decomposition*

$J_m + J_m^T = V_m \Lambda_m V_m^{-1}$

where Λ_m is the diagonal matrix containing all eigenvalues of $J_m + J_m^T$ and V_m is the unitary matrix containing all eigenvectors of $J_m + J_m^T$ corresponding to those eigenvalues as columns.

The using the results from [**9**] *the decomposition of the anti symmetric variant, $J_m - J_m^T$, is found to be:*

$$J_m - J_m^T = Z_m V_m \sqrt{-1} \Lambda_m (Z V_m)^{-1} \tag{3.9}$$

With

$$Z_m := \operatorname{diag}\left(\underset{i=1^m}{\boxed{\downarrow}} \left(\sqrt{-1}\right)^{i-1} \right) \tag{3.10}$$

4. Kronecker product

DEFINITION 28. *The **Kronecker product** between a $n \times m$-matrix A and a $p \times q$-matrix B is denoted by $A \otimes B$ and is defined as the $np \times mq$ block matrix:*

$$A \otimes B := \overset{n}{\underset{i=1}{\boxed{\Rightarrow}}} \overset{m}{\underset{j=1}{\boxed{\Downarrow}}} a_{(i,j)} B = \begin{pmatrix} a_{(1,1)}B & \dots & a_{(1,n)}B \\ \vdots & \ddots & \vdots \\ a_{(m,1)}B & \dots & a_{(m,n)}B \end{pmatrix}$$

LEMMA 11. $I_m \otimes I_p = I_{mp}$

LEMMA 12. $\lambda(A \otimes B) = \{\lambda_A \lambda_B \mid \lambda_A \in \lambda(A), \lambda_B \in \lambda(B)\}$

LEMMA 13. $(A \otimes B)_{i,j} = A_{i \bmod m, j \bmod n} B_{\lfloor \frac{i}{m} \rfloor, \lfloor \frac{j}{n} \rfloor}$

Matrix Norm of Kronecker Product

$$\begin{align} ||(B \otimes A)|| &= \max_{||x||=1} ||(B \otimes A)\mathbf{x}|| \tag{4.1} \\ &= \max_{||x||=1} ||\mathrm{vec}(A \ \mathrm{matrix}(\mathbf{x}) B^T)|| \tag{4.2} \\ &= \max_{||x||=1} ||A \ \mathrm{matrix}(\mathbf{x}) B^T|| \tag{4.3} \\ &\leq \max_{||x||=1} ||A|| \, || \ \mathrm{matrix}(\mathbf{x})|| \, ||B^T|| \tag{4.4} \\ &= ||A|| \, ||B|| \tag{4.5} \end{align}$$

DEFINITION 29. *The **Sequence Kronecker Product***

$$\bigotimes_{k=1}^{d} A_k = A_1 \otimes \ldots \otimes A_d$$

5. Testcase analysis

5.1. Testcase 1: Heat equation. See Appendix 3.1 for details on the formulation of the Heat equation.

$$d\frac{\partial u}{\partial t} = \Delta u \tag{5.1}$$

$$L := d\frac{\partial}{\partial t} - \Delta \tag{5.2}$$

$$:= d\frac{\partial}{\partial t} - \sum_{k=1}^{d} \frac{\partial^2}{\partial x_k^2} \tag{5.3}$$

5.1.1. *FTCS.* First using Forward in Time, Central in Space (FTCS) discretization of L:

$$S^n := \sigma_{Ft} - \sum_{k=1}^{d} \sigma^2_{Cx_k^{\frac{1}{2}}} \tag{5.4}$$

or in stencil form:

$$S^n = \{(-h_t^{-1}, \mathbf{0}), (h_t^{-1}, \mathbf{e}_{d+1})\} \cup \bigcup_{k=1}^{d} \{(-h_x^{-2}, -\mathbf{e}_k), (2h_x^{-2}, \mathbf{0})(-h_x^{-2}, \mathbf{e}_k)\} \tag{5.5}$$

$$= \{(2dh_x^{-2} - h_t^{-1}, \mathbf{0}), (h_t^{-1}, \mathbf{e}_{d+1})\} \cup \bigcup_{k=1}^{d} \{(-h_x^{-2}, -\mathbf{e}_k), (-h_x^{-2}, \mathbf{e}_k)\} \tag{5.6}$$

Using the local truncation error estimates found in 2.1 the local truncation error of the FTCS discretization of L can be estimated as:

$$\tau = \mathcal{O}(h_t) + \sum_{k=1}^{d_x} \mathcal{O}(h_x^2) \tag{5.7}$$

$$= \mathcal{O}(h_t) + \mathcal{O}(h_x^2) \tag{5.8}$$

Using the matrix representation the stability of the arising system can be analyzed. A FT discretization with zero boundary terms will result in a two level stencil which enabled the use of two level approach described in 5.5.1.

Matrices $A^n_{S^n}$ and $B^n_{S^n}$ are given by

$$A^n_{S^n} = h_t^{-1}Q^{\mathbf{0}} \tag{5.9}$$

$$B^n_{S^n} = (h_t^{-1} - 2dh_x^{-2})Q^{\mathbf{0}} + h_x^{-2}\sum_{k=1}^{d} Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k} \tag{5.10}$$

$$\tag{5.11}$$

The eigenvalues of these matrices are then derived:

$$\lambda(A^n_{S^n}) = \lambda(h_t^{-1}Q^{\mathbf{0}}) = \lambda(h_t^{-1}I) = \{h_t^{-1}\} \tag{5.12}$$

and

$$\begin{aligned}
\lambda(B^n_{S^n}) &= \lambda\left((h_t^{-1} - 2dh_x^{-2})Q^{\mathbf{0}} + h_x^{-2}\sum_{k=1}^{d} Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}\right) \\
&= \lambda\left((h_t^{-1} - 2dh_x^{-2})I + h_x^{-2}\sum_{k=1}^{d} Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}\right) \qquad (5.13)
\end{aligned}$$

$Q^{\mathbf{0}} = I$ is so:

$$= h_t^{-1} - 2dh_x^{-2} + h_x^{-2}\lambda\left(\sum_{k=1}^{d} I_{m^{k-1}} \otimes (J_m^T + J_m) \otimes I_{m^{d-k}}\right) \tag{5.14}$$

Using the matrix norm to estimate the eigenvalues:

$$\subseteq h_t^{-1} - 2dh_x^{-2} + h_x^{-2}\left\|\sum_{k=1}^{d} I_{m^{k-1}} \otimes (J_m^T + J_m) \otimes I_{m^{d-k}}\right\|\mathbb{C}_{\leq 1} \tag{5.15}$$

This is a real symmetric matrix so its eigenvalues must be real:

$$= h_t^{-1} - 2dh_x^{-2} + h_x^{-2}\left\|\sum_{k=1}^{d} I_{m^{k-1}} \otimes (J_m^T + J_m) \otimes I_{m^{d-k}}\right\|[-1,1] \tag{5.16}$$

Using Matrix norm estimates:

$$\subseteq h_t^{-1} - 2dh_x^{-2} + dh_x^{-2}\left\|J_m^T + J_m\right\|[-1,1] \tag{5.17}$$

$$= h_t^{-1} - 2dh_x^{-2} + 2dh_x^{-2}[-1,1] \tag{5.18}$$

$$= h_t^{-1} - 2dh_x^{-2}[0,2] \tag{5.19}$$

To uphold stability according to (5.34) the modulus of the eigenvalues must be restricted to $(0,1)$ which yields the following restraints:

$$||\lambda((A^n_{S^n})^{-1}B^n_{S^n})|| \subset (0,1) \tag{5.20}$$

$$\Leftrightarrow ||1 + h_x^{-2}h_t[-4d,0]|| \subset (0,1) \tag{5.21}$$

Since this is real value this reduces to:

$$\Leftrightarrow 1 + h_x^{-2} h_t[-4d, 0] \subset (-1, 1) \tag{5.22}$$

$$\Leftrightarrow h_x^{-2} h_t[-4d, 0] \subset (-2, 0) \tag{5.23}$$

$$\Leftrightarrow h_x^{-2} h_t \in (0, \frac{1}{2d}) \tag{5.24}$$

$$\Leftrightarrow h_t < \frac{h_x^2}{2d} \tag{5.25}$$

The FTCS discretization of the diffusion PDE is conditionally stable. For a given spatial step size, h_x, the stable time step size, h_t, is limited and vice versa. This is consistent with the results for 1 and 4 dimensions in [**10**].

5.1.2. *BTCS.* Now using a Backward in Time, Central in Space (BTCS) discretization of L:

$$S^n := \sigma_{Bt} - \sum_{k=1}^{d} \sigma^2_{Cx_k^{\frac{1}{2}}}$$

or in stencil form:

$$\begin{aligned} S^n &= \{(h_t^{-1}, \mathbf{0}), (-h_t^{-1}, -\mathbf{e}_{d+1})\} \cup \bigcup_{k=1}^{d} \{(-h_x^{-2}, -\mathbf{e}_k), (2h_x^{-2}, \mathbf{0})(-h_x^{-2}, \mathbf{e}_k)\} \\ &= \{(2dh_x^{-2} + h_t^{-1}, \mathbf{0}), (-h_t^{-1}, -\mathbf{e}_{d+1})\} \cup \bigcup_{k=1}^{d} \{(-h_x^{-2}, -\mathbf{e}_k), (-h_x^{-2}, \mathbf{e}_k)\} \end{aligned}$$

Using the local truncation error estimates found in 2.1 the local truncation error of the BTCS discretization of L can be estimated as:

$$\tau = \mathcal{O}(h_t) + \sum_{k=1}^{d_x} \mathcal{O}(h_x^2) \tag{5.26}$$

$$= \mathcal{O}(h_t) + \mathcal{O}(h_x^2) \tag{5.27}$$

Using the matrix representation the stability of the arising system can be analyzed. A BT discretization with zero boundary terms will again result in a two level stencil which enabled the use of two level approach described in 5.5.1.

Matrices $A^n_{S^n}$ and $B^n_{S^n}$ are given by

$$\begin{aligned} A^n_{S^n} &= (2dh_x^{-2} + h_t^{-1})Q^{\mathbf{0}} - h_x^{-2} \sum_{k=1}^{d} Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k} \\ B^n_{S^n} &= h_t^{-1} Q^{\mathbf{0}} \end{aligned}$$

The eigenvalues of these matrices are then derived:

$$\lambda(A^n_{S^n}) = \lambda\left((2dh_x^{-2} + h_t^{-1})Q^{\mathbf{0}} - h_x^{-2} \sum_{k=1}^{d} Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}\right)$$

Using the steps from the previous section (FTCS):

$$\subseteq \quad 2dh_x^{-2} + h_t^{-1} - 2h_x^{-2}[-1,1] \tag{5.28}$$

$$= \quad h_t^{-1} + h_x^{-2}[0,4d] \tag{5.29}$$

and

$$\lambda_{\max}(B^n_{S^n}) \quad = \quad \lambda_{\max}(h_t^{-1}Q^{\mathbf{0}}) = \lambda_{\max}(h_t^{-1}I) = \{h_t^{-1}\} \tag{5.30}$$

The maximum eigenvalues of the rearranged matrix $(A^n_{S^n})^{-1}(B^n_{S^n})$ are confined to:

$$\lambda((A^n_{S^n})^{-1}(B^n_{S^n})) \quad \subseteq \quad \left(h_t^{-1} + h_x^{-2}[0,4d]\right)^{-1}(h_t^{-1}) \tag{5.31}$$

$$= \quad \left(1 + h_t h_x^{-2}[0,4d]\right)^{-1} \tag{5.32}$$

Since h_x, h_t and d are all real and strictly positive $\lambda((A^n_{S^n})^{-1}B^n_{S^n} \subset (0,1)$ and thus by (5.34) the BTCS approximation of the diffusion PDE is unconditionally stable. This is consistent with the results for 1 and 4 dimensions in [**10**].

5.2. Testcase 2 : Implosion equation. See Appendix 3.2 for details on the formulation of the Implosion equation. The implosion equation is non linear in $x_1, \ldots, x_d$ but still linear in u

$$\frac{\partial u}{\partial t} = -\mathbf{x} \cdot \nabla u \tag{5.33}$$

where ∇ is by default defined as the spatial gradient: $\nabla_{\mathbf{x}} = \boxed{\downarrow}_{k=1}^{d_{\mathbf{x}}} \frac{\partial}{\partial x_k}$. The exact solution to the implosion problem is given by:

$$u(x,t) = e^{-de^t ||x||_2^2} \tag{5.34}$$

$$L \quad := \quad \frac{\partial}{\partial t} + \mathbf{x}(\cdot) \cdot \nabla \tag{5.35}$$

$$:= \quad \frac{\partial}{\partial t} + \sum_{k=1}^{d} x_k(\cdot) \frac{\partial}{\partial x_k} \tag{5.36}$$

5.2.1. *FTCS.* When the boundaries of the domain are sufficiently far from the origin the solution will be near zero on those boundaries.

$$S^n = \sigma_{Ft} + \sum_{k=1}^{d} \operatorname{diag}(\mathbf{x}_k) S_{Cx_k} \tag{5.37}$$

or in stencil form:

$$S^n = \{(-h_t^{-1}, \mathbf{0}), (h_t^{-1}, \mathbf{e}_{d+1})\} \cup \bigcup_{k=1}^{d} \{((2h_x)^{-1}x_k(\cdot), \mathbf{e}_k), (-(2h_x)^{-1}x_k(\cdot), -\mathbf{e}_k)\}$$

Using the local truncation error estimates found in 2.1 the local truncation error of the FTCS discretization of L can be estimated as:

$$\tau = \mathcal{O}(h_t) + \sum_{k=1}^{d} \operatorname{diag}(\mathbf{x}_k)\mathcal{O}(h_x^2) \tag{5.38}$$

$$= \mathcal{O}(h_t) + \mathcal{O}(h_x^2) \tag{5.39}$$

Using the matrix representation the stability of the arising system can be analyzed. A FT discretization with (forced) zero boundary terms will result in a two level stencil which enabled the use of two level approach described in 5.5.1.

$A^n_{S^n}$ and $B^n_{S^n}$ are given by:

$$\begin{aligned} A^n_{S^n} &= h_t^{-1} Q^{\mathbf{0}} \\ B^n_{S^n} &= h_t^{-1} Q^{\mathbf{0}} - (2h_x)^{-1} \sum_{k=1}^{d} \operatorname{diag}(\mathbf{x}_k) \left(Q^{\mathbf{e}_k} - Q^{-\mathbf{e}_k} \right) \end{aligned}$$

The eigenvalues can be estimated by:

$$\lambda(A^n_{S^n}) = h_t^{-1}\lambda(Q^{\mathbf{0}}) = \lambda(h_t^{-1} I_{m^d}) = \{h_t^{-1}\} \tag{5.40}$$

Define $X_k := \operatorname{diag}\left(\boxed{\downarrow}_{i=1}^{m_1} - (\vec{q}_{\min})_k + h_{x_k} i \right)$ such that $\operatorname{diag}(\mathbf{x}_k) = I_{m^{k-1}} \otimes X_k \otimes I_{m^{d-k}}$.

$$\lambda(B^n_{S^n}) = \lambda\left(h_t^{-1} Q^{\mathbf{0}} - h_x^{-1} \sum_{k=1}^{d} \operatorname{diag}(\mathbf{x}_k) \left(Q^{\mathbf{e}_k} - Q^{-\mathbf{e}_k} \right) \right)$$

$Q^{\mathbf{0}} = I$ is so:

$$= h_t^{-1} - (2h_x)^{-1} \lambda\left(\sum_{k=1}^{d} \operatorname{diag}(\mathbf{x}_k) \left(Q^{\mathbf{e}_k} - Q^{-\mathbf{e}_k} \right) \right) \tag{5.41}$$

Using the matrix norm to estimate the eigenvalues:

$$\subseteq h_t^{-1} - (2h_x)^{-1} \left\| \sum_{k=1}^{d} \operatorname{diag}(\mathbf{x}_k) \left(Q^{\mathbf{e}_k} - Q^{-\mathbf{e}_k} \right) \right\| \mathbb{C}_{\leq 1} \tag{5.42}$$

$$= h_t^{-1} - (2h_x)^{-1} \left\| \sum_{k=1}^{d} \operatorname{diag}(\mathbf{x}_k) \left(I_{m^{k-1}} \otimes (J_m - J_m^T) \otimes I_{m^{d-k}} \right) \right\| \mathbb{C}_{\leq 1} \tag{5.43}$$

$$= h_t^{-1} - (2h_x)^{-1} \left\| \sum_{k=1}^{d} \left(I_{m^{k-1}} \otimes X_k (J_m - J_m^T) \otimes I_{m^{d-k}} \right) \right\| \mathbb{C}_{\leq 1} \tag{5.44}$$

This a real anti-symmetric matrix so its eigenvalues must be purely imaginary:

$$= h_t^{-1} - (2h_x)^{-1} \sqrt{-1} \left\| \sum_{k=1}^{d} \left(I_{m^{k-1}} \otimes X_k (J_m - J_m^T) \otimes I_{m^{d-k}} \right) \right\| [-1, 1] \tag{5.45}$$

Using Matrix norm estimates:

$$= h_t^{-1} - (2h_x)^{-1} \sqrt{-1} dz \left\| (J_m - J_m^T) \right\| [-1, 1] \tag{5.46}$$

With $z := \max_{1\leq i\leq m, 1\leq k\leq d} |(X_k)_i|$.

$$= h_t^{-1} - h_x^{-1}\sqrt{-1}dz[-1,1] \tag{5.47}$$

To uphold stability according to (5.34) the moduli of the eigenvalues must be restricted to $(-1,1)$ which yields the following restraints:

$$\begin{aligned}
&||\lambda((A^n_{S^n})^{-1}B^n_{S^n})|| \subset (0,1) && (5.48)\\
\Leftrightarrow\ &||1 - h_t h_x^{-1}\sqrt{-1}dz[-1,1]|| \subset (0,1) && (5.49)\\
\Leftrightarrow\ &\sqrt{1+(-h_t h_x^{-1}dz[-1,1])^2} \subset (0,1) && (5.50)\\
\Leftrightarrow\ &\sqrt{1+(h_t h_x^{-1}dz)^2[0,1]} \subset (0,1) && (5.51)\\
\Leftrightarrow\ &1+(h_t h_x^{-1}dz)^2[0,1] \subset (0,1) && (5.52)\\
& && (5.53)
\end{aligned}$$

From observing that $(h_t d(2h_x)^{-1}z > 0$ follows that the FTCS discretization of the implosion PDE is unconditionally unstable with respect to the estimates made.

5.2.2. *BTCS.*

$$S^n = \sigma_{Bt} + \sum_{k=1}^{d_x} \mathrm{diag}(\mathbf{x}_k) S_{Cx_k} \tag{5.54}$$

or in stencil form:

(5.55)

$$S^n = \{h_t^{-1}, \mathbf{0}), (-h_t^{-1}, -\mathbf{e}_{d+1})\} \cup \bigcup_{k=1}^{d} \{((2h_x)^{-1}x_k(\cdot), \mathbf{e}_k), (-(2h_x)^{-1}x_k(\cdot), -\mathbf{e}_k)\}$$

Using the local truncation error estimates found in 2.1 the local truncation error of the BTCS discretization of L can be estimated as:

$$\begin{aligned}
\tau &= \mathcal{O}(h_t) + \sum_{k=1}^{d_x} \mathrm{diag}(\mathbf{x}_k)\mathcal{O}(h_x^2) && (5.56)\\
&= \mathcal{O}(h_t) + \mathcal{O}(h_x^2) && (5.57)
\end{aligned}$$

Again assuming zero boundaries a BT discretization will also result in a two level stencil which enabled the use of approach described in 5.5.1.

$A^n_{S^n}$ and $B^n_{S^n}$ are given by:

$$\begin{aligned}
A^n_{S^n} &= h_t^{-1}Q^{\mathbf{0}} + (2h_x)^{-1}\sum_{k=1}^{d} \mathrm{diag}(\mathbf{x}_k)\left(Q^{\mathbf{e}_k} - Q^{-\mathbf{e}_k}\right) && (5.58)\\
B^n_{S^n} &= h_t^{-1}Q^{\mathbf{0}} && (5.59)
\end{aligned}$$

The eigenvalues can be estimated by:

$$\lambda(A^n_{S^n}) \quad = \quad \lambda\left(h_t^{-1}Q^{\mathbf{0}} + (2h_x)^{-1}\sum_{k=1}^{d}\operatorname{diag}(\mathbf{x}_k)\left(Q^{\mathbf{e}_k} - Q^{-\mathbf{e}_k}\right)\right) \tag{5.60}$$

Using the steps of the FTCS discretization:

$$\subseteq \quad h_t^{-1} + h_x^{-1}\sqrt{-1}zd[-1,1] \tag{5.61}$$

and

$$\lambda(B^n_{S^n}) \quad = \quad h_t^{-1}\lambda(Q^{\mathbf{0}}) = \lambda(h_t^{-1}I_{m^d}) = \{h_t^{-1}\} \tag{5.62}$$

To uphold stability according to (5.34) the moduli of the eigenvalues must be restricted to $[-1,1]$ which yields the following restraints:

$$\left\|\lambda\left((A^n_{S^n})^{-1}B^n_{S^n}\right)\right\| \quad \subset \quad (0,1) \tag{5.63}$$

$$\Leftrightarrow ||\frac{h_t^{-1}}{h_t^{-1} + h_x^{-1}\sqrt{-1}zd[-1,1]}|| \quad \subset \quad (0,1) \tag{5.64}$$

$$\Leftrightarrow ||\frac{1}{1 + h_t h_x^{-1}\sqrt{-1}zd[-1,1]}|| \quad \subset \quad (0,1) \tag{5.65}$$

$$\Leftrightarrow ||1 + h_t h_x^{-1}\sqrt{-1}zd[-1,1]|| \quad \subset \quad (1,\infty) \tag{5.66}$$

$$\Leftrightarrow \sqrt{1 + (h_t h_x^{-1}zd[-1,1])^2} \quad \subset \quad (1,\infty) \tag{5.67}$$

$$\Leftrightarrow \sqrt{1 + (h_t h_x^{-1}zd)^2[0,1]} \quad \subset \quad (1,\infty) \tag{5.68}$$

$$\Leftrightarrow 1 + (h_t h_x^{-1}zd)^2[0,1] \quad \subset \quad (1,\infty) \tag{5.69}$$

Since h_t, h_x, z and d are all reals the BTCS discretization of the implosion PDE is unconditionally stable.

5.3. Testcase 3 : Burgers' equation.

See Appendix 3.3 for details on the formulation of the Burgers' equation. Burgers' equation is non linear in u

$$\frac{\partial u}{\partial t} = -u\mathbf{1}\cdot\nabla u + \mu\Delta u \tag{5.70}$$

$$L \quad := \quad \frac{\partial}{\partial t} + \mathbf{u}(\cdot)\mathbf{1}\cdot\nabla - \mu\Delta \tag{5.71}$$

$$:= \quad \frac{\partial}{\partial t} + \sum_{k=1}^{d} u\frac{\partial}{\partial x_k} - \mu\frac{\partial^2}{\partial x_k^2} \tag{5.72}$$

5.3.1. *FTCS.* First using Forward in Time, Central in Space (FTCS) discretization of L:

$$S^n := \sigma_{Ft} + \sum_{k=1}^{d} u(\cdot)\sigma_{Cx_k} - \mu\sigma^2_{Cx_k^{\frac{1}{2}}} \tag{5.73}$$

or in stencil form:

$$\begin{aligned} S^n &= \{(-h_t^{-1}, \mathbf{0}), (h_t^{-1}, \mathbf{e}_{d+1})\} \cup \\ &\bigcup_{k=1}^{d} \{(-u(\cdot)(2h_x)^{-1} - \mu h_x^{-2}, -\mathbf{e}_k), (\mu 2h_x^{-2}, \mathbf{0}), (u(\cdot)(2h_x)^{-1} - \mu h_x^{-2}, \mathbf{e}_k)\} \quad (5.74) \\ &= \{(2\mu d h_x^{-2} - h_t^{-1}, \mathbf{0}), (h_t^{-1}, \mathbf{e}_{d+1})\} \cup \quad (5.75) \\ &\bigcup_{k=1}^{d} \{(-u(\cdot)(2h_x)^{-1} - \mu h_x^{-2}, -\mathbf{e}_k), (u(\cdot)(2h_x)^{-1} - \mu h_x^{-2}, \mathbf{e}_k)\} \quad (5.76) \end{aligned}$$

Using the local truncation error estimates found in 2.1 the local truncation error of the FTCS discretization of L can be estimated as:

$$\tau = \mathcal{O}(h_t) + \sum_{k=1}^{d_x} \operatorname{diag}(\mathbf{x}_k)\mathcal{O}(h_x^2) \operatorname{diag}(\mathbf{u}^{n-1})\mathcal{O}(h_x^2) \quad (5.77)$$

$$= \mathcal{O}(h_t) + \mathcal{O}(h_x^2) \quad (5.78)$$

Using the matrix representation the stability of the arising system can be analyzed. A FT discretization results in a two level stencil when the perturbations in the boundary terms are discarded which enabled the use of two level approach described in 5.5.1.

Matrices $A_{S^n}^n$ and $B_{S^n}^n$ are given by

$$\begin{aligned} A_{S^n}^n &= h_t^{-1} Q^{\mathbf{0}} \\ B_{S^n}^n &= (h_t^{-1} - 2\mu d h_x^{-2}) Q^{\mathbf{0}} \\ &\quad + \sum_{k=1}^{d} (\mu h_x^{-2} - U^{n-1}(2h_x)^{-1}) Q^{-\mathbf{e}_k} + (\mu h_x^{-2} + U^{n-1}(2h_x)^{-1}) Q^{\mathbf{e}_k} \\ &= (h_t^{-1} - 2\mu d h_x^{-2}) Q^{\mathbf{0}} \\ &\quad + \sum_{k=1}^{d} \mu h_x^{-2} (Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}) + U^{n-1}(2h_x)^{-1}(Q^{-\mathbf{e}_k} - Q^{\mathbf{e}_k}) \end{aligned}$$

With $U^n := \operatorname{diag}(\mathbf{u}^n)$. The eigenvalues of these matrices are then derived:

$$\lambda(A_{S^n}^n) = \lambda(h_t^{-1} Q^{\mathbf{0}}) = \lambda(h_t^{-1} I) = \{h_t^{-1}\} \quad (5.79)$$

and

$$\begin{aligned} &\lambda(B_{S^n}^n) = \\ &\lambda\left((h_t^{-1} - 2\mu d h_x^{-2}) Q^{\mathbf{0}} + \sum_{k=1}^{d} \mu h_x^{-2} (Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}) + U^{n-1}(2h_x)^{-1}(Q^{-\mathbf{e}_k} - Q^{\mathbf{e}_k}) \right) \end{aligned}$$

Using that $Q^{\mathbf{0}} = I$, so:

$$= h_t^{-1} - 2\mu d h_x^{-2} + \lambda\left(\sum_{k=1}^{d} \mu h_x^{-2} (Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}) + U^{n-1}(2h_x)^{-1}(Q^{-\mathbf{e}_k} - Q^{\mathbf{e}_k}) \right)$$

Using the matrix norm to estimate the eigenvalues:

$$= \quad h_t^{-1} - 2\mu d h_x^{-2} + \left\| \sum_{k=1}^{d} \mu h_x^{-2}(Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}) + U^{n-1}(2h_x)^{-1}(Q^{-\mathbf{e}_k} - Q^{\mathbf{e}_k}) \right\|$$

Using Matrix norm estimates:

$$\begin{aligned} &\subseteq \quad h_t^{-1} - 2\mu d h_x^{-2} + d\left(\mu h_x^{-2}\left\|(J^T + J)\right\| + (2h_x)^{-1}\left\|U^{n+1}\right\|\left\|(J^T - J)\right\|\right)\mathbb{C}_{\leq 1} \\ &= \quad h_t^{-1} - 2\mu d h_x^{-2} + d\left(\mu 2 h_x^{-2} + h_x^{-1} v\right)\mathbb{C}_{\leq 1} \end{aligned}$$

with $v := \max_{i \in \mathcal{T}_{n-1}} |\mathbf{u}_i|$

This is a very crude estimate which results in overestimation and unconditional unstability. $J^T + J$ is symmetric and thereforeit has only real eigenvalues. $J^T - J$ on the other hand is anti-symmetric and thereforehas only purely imaginary eigenvalues. The modulo of the eigenvalues of their linear combination can be estimated by using the the absolute value of the same linear combination of the matrix norms.

Empirical results actually show that the matrix $\alpha J(^T{+}J) + \beta(J^T - J)$ has either real or purely imaginary eigenvalues for $\alpha, \beta \neq 0$. Stability is possible. This is explored as follows:

$$\begin{aligned} &= \quad h_t^{-1} - 2\mu d h_x^{-2} + \lambda\left(\sum_{k=1}^{d} \mu h_x^{-2}(J^T + J) + U^{n-1}(2h_x)^{-1}\sqrt{-1}(J^T - J)\right) \\ &= \quad h_t^{-1} - 2\mu d h_x^{-2} + \lambda\left(\sum_{k=1}^{d} (\mu h_x^{-2} - U^{n-1}(2h_x)^{-1})J + (\mu h_x^{-2} + U^{n-1}(2h_x)^{-1})J^T\right) \end{aligned}$$

Assuming $u \equiv v$ constant, such that $U^{n-1} = vI$, the following estimate can be made. This is an unfounded approach that needs rigorous analysis but seems to provide helpful estimates.

$$\begin{aligned} &= \quad h_t^{-1} - 2\mu d h_x^{-2} + \sum_{k=1}^{d} \lambda\left((\mu h_x^{-2} - v(2h_x)^{-1})J + (\mu h_x^{-2} + v(2h_x)^{-1})J^T\right) \\ &\subseteq \quad h_t^{-1} - 2\mu d h_x^{-2} + d\sqrt{(\mu h_x^{-2} - v(2h_x)^{-1})(\mu h_x^{-2} + v(2h_x)^{-1})2}[-1,1] \\ &= \quad h_t^{-1} - 2\mu d h_x^{-2} + d\sqrt{((\mu h_x^{-2})^2 - (v(2h_x)^{-1})^2)2}[-1,1] \end{aligned}$$

To uphold stability according to (5.34) the moduli of the eigenvalues must be restricted to $(0, 1)$ which yields the following restraints:

$$||\lambda((A_{S^n}^n)^{-1}B_{S^n}^n)|| \quad \subset \quad (0,1) \tag{5.80}$$

$$\Leftrightarrow ||1 - 2\mu d h_t h_x^{-2} + h_t d\sqrt{((\mu h_x^{-2})^2 - (v(2h_x)^{-1})^2)2}[-1,1]|| \quad \subset \quad (0,1) \tag{5.81}$$

(5.82)

This represents a horizontal or vertical segment centered on $1 - 2\mu d h_t h_x^{-2}$ that should fit in the complex unit disk. It follows that the FTCS discretization of Burgers' PDE is conditionally stable.

5.3.2. *BTCS.* Using Backward in Time, Central in Space (BTCS) discretization of L:

$$S^n := \sigma_{Bt} + \sum_{k=1}^{d} u(\cdot)\sigma_{Cx_k} - \mu\sigma^2_{Cx_k^{\frac{1}{2}}} \tag{5.83}$$

or in stencil form:

$$\begin{aligned}
S^n &= \{(h_t^{-1}, \mathbf{0}), (-h_t^{-1}, -\mathbf{e}_{d+1})\} \cup \\
&\quad \bigcup_{k=1}^{d} \{(-u(\cdot)(2h_x)^{-1} - \mu h_x^{-2}, -\mathbf{e}_k), (\mu 2h_x^{-2}, \mathbf{0}), (u(\cdot)(2h_x)^{-1} - \mu h_x^{-2}, \mathbf{e}_k)\} && (5.84)\\
&= \{(2\mu d h_x^{-2} + h_t^{-1}, \mathbf{0}), (-h_t^{-1}, -\mathbf{e}_{d+1})\} \cup && (5.85)\\
&\quad \bigcup_{k=1}^{d} \{(-u(\cdot)(2h_x)^{-1} - \mu h_x^{-2}, -\mathbf{e}_k), (u(\cdot)(2h_x)^{-1} - \mu h_x^{-2}, \mathbf{e}_k)\} && (5.86)
\end{aligned}$$

Using the local truncation error estimates found in 2.1 the local truncation error of the BTCS discretization of L can be estimated as:

$$\begin{aligned}
\tau &= \mathcal{O}(h_t) + \sum_{k=1}^{d_x} \operatorname{diag}(\mathbf{x}_k)\mathcal{O}(h_x^2)\operatorname{diag}(\mathbf{u}^{n-1})\mathcal{O}(h_x^2) && (5.87)\\
&= \mathcal{O}(h_t) + \mathcal{O}(h_x^2) && (5.88)
\end{aligned}$$

Using the matrix representation the stability of the arising system can be analyzed. A BT discretization results in a two level stencil when the perturbations in the boundary terms are discarded which enabled the use of two level approach described in 5.5.1.

Matrices $A^n_{S^n}$ and $B^n_{S^n}$ are given by

$$\begin{aligned}
A^n_{S^n} &= (h_t^{-1} + 2\mu d h_x^{-2})Q^{\mathbf{0}} \\
&\quad - \sum_{k=1}^{d} (\mu h_x^{-2} - U^{n-1}(2h_x)^{-1})Q^{-\mathbf{e}_k} + (\mu h_x^{-2} + U^{n-1}(2h_x)^{-1})Q^{\mathbf{e}_k} \\
&= (h_t^{-1} + 2\mu d h_x^{-2})Q^{\mathbf{0}} \\
&\quad - \sum_{k=1}^{d} \mu h_x^{-2}(Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}) + U^{n-1}(2h_x)^{-1}(Q^{-\mathbf{e}_k} - Q^{\mathbf{e}_k}) \\
B^n_{S^n} &= h_t^{-1}Q^{\mathbf{0}}
\end{aligned}$$

With $U^n := \mathrm{diag}(\mathbf{u}^n)$. The eigenvalues of these matrices are then derived:

$$\lambda(A^n_{S^n}) =$$
$$\lambda\left((h_t^{-1} + 2\mu d h_x^{-2})Q^{\mathbf{0}} - \sum_{k=1}^{d} \mu h_x^{-2}(Q^{-\mathbf{e}_k} + Q^{\mathbf{e}_k}) + U^{n-1}(2h_x)^{-1}(Q^{-\mathbf{e}_k} - Q^{\mathbf{e}_k})\right)$$

Using the steps from the FTCS discretization:

$$\subseteq \quad h_t^{-1} + 2\mu d h_x^{-2} - d(\mu 2 h_x^{-2} + h_x^{-1} v)\mathbb{C}_{\leq 1}$$

with $v := \max_{i \in \mathcal{T}_{n-1}} |\mathbf{u}_i|$

and

$$\lambda(B^n_{S^n}) \;=\; \lambda(h_t^{-1}Q^{\mathbf{0}}) = \lambda(h_t^{-1}I) = \{h_t^{-1}\} \tag{5.89}$$

Just as with the FTCS case this is a very crude estimate which results in overestimation and unconditional unstability. $J^T + J$ is symmetric and thereforeit has only real eigenvalues. $J^T - J$ on the other hand is anti-symmetric and thereforehas only purely imaginary eigenvalues. The modulo of the eigenvalues of their linear combination can be estimated by using the the absolute value of the same linear combination of the matrix norms.

Empirical results actually show that the matrix $\alpha J(^T + J) + \beta(J^T - J)$ has either real or purely imaginary eigenvalues for $\alpha, \beta \neq 0$. Stability is possible. This is explored as follows:

$$\begin{aligned}
&= \quad h_t^{-1} - 2\mu d h_x^{-2} - \lambda\left(\sum_{k=1}^{d} \mu h_x^{-2}(J^T + J) + U^{n-1}(2h_x)^{-1}\sqrt{-1}(J^T - J)\right) \\
&= \quad h_t^{-1} + 2\mu d h_x^{-2} - \lambda\left(\sum_{k=1}^{d} (\mu h_x^{-2} - U^{n-1}(2h_x)^{-1})J + (\mu h_x^{-2} + U^{n-1}(2h_x)^{-1})J^T\right)
\end{aligned}$$

Assuming $u \equiv v$ constant, such that $U^{n-1} = vI$, the following estimate can be made. This is an unfounded approach that needs rigorous analysis but seems to provide helpful estimates.

$$\begin{aligned}
&= \quad h_t^{-1} + 2\mu d h_x^{-2} - \sum_{k=1}^{d} \lambda\left((\mu h_x^{-2} - v(2h_x)^{-1})J + (\mu h_x^{-2} + v(2h_x)^{-1})J^T\right) \\
&\subseteq \quad h_t^{-1} + 2\mu d h_x^{-2} - d\sqrt{(\mu h_x^{-2} - v(2h_x)^{-1})(\mu h_x^{-2} + v(2h_x)^{-1})}2[-1, 1] \\
&= \quad h_t^{-1} + 2\mu d h_x^{-2} - d\sqrt{((\mu h_x^{-2})^2 - (v(2h_x)^{-1})^2)}2[-1, 1]
\end{aligned}$$

To uphold stability according to (5.34) the moduli of the eigenvalues must be restricted to $(0,1)$ which yields the following restraints:

$$||\lambda((A^n_{S^n})^{-1}B^n_{S^n})|| \subset (0,1) \tag{5.90}$$

$$\Leftrightarrow ||\frac{h_t^{-1}}{h_t^{-1}+2\mu d h_x^{-2} - d\sqrt{((\mu h_x^{-2})^2-(v(2h_x)^{-1})^2)}2[-1,1]}|| \subset (0,1) \tag{5.91}$$

$$\Leftrightarrow ||1+2\mu d h_t h_x^{-2} - h_t d\sqrt{((\mu h_x^{-2})^2-(v(2h_x)^{-1})^2)}2[-1,1]|| \subset (1,\infty) \tag{5.92}$$

This represents a horizontal or vertical segment centered on $1+2\mu d h_t h_x^{-2}$ that should fit in the complex unit disk. Since the segment is of length $2h_t d\sqrt{((\mu h_x^{-2})^2-(v(2h_x)^{-1})^2)}$ and thus shorter than $2\mu d h_t h_x^{-2}$ it follows that the BTCS discretization of Burgers' PDE is unconditionally stable.

6. Notation

6.1. Font use for variables. As a rule of thumb the following font use is used to distinguish symbol properties. Note that there are various exemptions.

$a, b, c, \ldots$	: Functions and scalar values.
$A, B, C, \ldots$	: Matrices and equation defining operators
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$	: Vectors
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$	: Sets

For the i-th element of a vector of scalars $\mathbf{a}$ is denoted using a subscript as a_i. For a vector $\mathbf{a}_k$ that already has subscripts this is solved by noting $(\mathbf{a}_k)_i$.

6.2. Domain.

$d_\mathbf{a}$	$:= \lvert\mathbf{a}\rvert$ i.e. $\mathbf{a} \in \mathbb{R}^{d_\mathbf{a}}$
d	$:= d_\mathbf{x}$ Default dimension.
Ω	$:= \Omega_\mathbf{x}$ Default domain.
$\Omega_\mathbf{x}$	$\subset \mathbb{R}^{d_\mathbf{x}}$: Space domain
T	: Upper time boundary.
$\Omega_\mathbf{q}$	$:= \Omega_\mathbf{x} \times [0,T] \subset \mathbb{R}^{d_\mathbf{q}}$ Space-time domain
k	$:\in \{1,\ldots,d\}$ dimensional index.
$\mathbf{x}$	$:= (x_1,\ldots,x_d)$ spatial coordinates.
t	: Temporal coordinate.
$\mathbf{q}$	$:= (q_1,\ldots,q_{d_\mathbf{q}}) := (x_1,\ldots,x_d,t)$ Generalized spatial temporal coordinates.
$\partial_{k,l}$	: Boundary operator: the l-th k-dimensional boundary.

l_k	$:$ Number of k-dimensional boundaries.
l	$:\in \{1, \ldots, l_k\}$ Boundary index.
int	$:= \sum_{l=1}^{l_d} \partial_{d,l}$ Interior operator.
u	$: \Omega \to \mathbb{R}$ Solution function.
f	$: \Omega \to \mathbb{R}$ Right hand function.

6.3. Differential operators.

$\frac{\partial}{\partial q_k}$	$:$ atomic partial differential operator
$\frac{\partial^n}{\partial q_{k_1} \ldots \partial q_{k_n}}$	$:= \frac{\partial}{\partial q_{k_1}} \circ \ldots \circ \frac{\partial}{\partial q_{k_n}}$ higher order partial differential operator.
$\mathcal{D}_{\mathbf{v}}$	$:$ atomic directional differential operator.
$\Delta_{\mathbf{a}}$	$:= \sum_{k=1}^{\lvert\mathbf{a}\rvert} \partial_{a_k a_k}$ Laplace operator.
$\nabla_{\mathbf{a}}$	$:= \boxed{\downarrow}_{k=1}^{\lvert\mathbf{a}\rvert} \partial_{a_k}$ Gradient operator.
Δ	$:= \Delta_{\mathbf{x}}$ Default Laplace operator.
∇	$:= \nabla_{\mathbf{x}}$ Default gradient operator.
L	$:$ Linear operator

6.4. Discretization.

6.4.1. *General Discretization.*

W	$:$ Discretization of Ω.
$\mathcal{I}$	$:$ Node index set
i	$:\in \mathcal{I}$ Node index
$\partial_{k,l}\mathcal{I}$	$:$ Boundary node index set
$\mathbf{q_i}$	$:$ Nodes
$\mathcal{Q}$	$:$ Node set
$\partial_{k,l}\mathcal{Q}$	$:$ Boundary node set
$\mathcal{J}_i$	$:$ Adjacency label set
j	$:\in \mathcal{J}_i$ Adjacency label
$\mathcal{N}$	$:$ Adjacency / Neighborhood set
ψ	$: \mathcal{I} \to \mathbb{R}$ Nodal function
Δ_i	$: \mathcal{J}_i \to \mathcal{N}$ Adjacency operator
$\mathbf{r_{i,j}}$	$:= \mathbf{q}_{i+\Delta j} - \mathbf{q}_i$ Relative neighborhood positions

$h_j(i)$	: Pseudo norm of $\mathbf{r}_{i,j}$
R_i	$:= \boxed{\rightarrow}_{j\in\mathcal{J}} \frac{\mathbf{r}_{i,j}}{h_j(i)}$ Transformation matrix.

6.4.2. *Traversal.*

N	: Number of traversal levels.
n	$:\in \{1,\dots,N\}$ traversal index.
$\mathcal{T}_n$	: n-th traversal level.

6.4.3. *Uniform Discretization.*

$\mathbf{m}$	$:= (m_1,\dots,m_d)$ Grid dimensions
$\mathbf{h}$	$:= (h_1,\dots,h_d)$ Grid cell dimensions
H	$:= \mathrm{diag}(\mathbf{h})$ Grid cell dimension matrix
$\mathbf{m}^k$	$:= m_1 \cdot \ldots \cdot m_k$ Vector product

6.4.4. *Function Discretization.*

$\mathbf{f}^{\mathcal{A}}$	$:= \boxed{\downarrow}_{i\in\mathcal{A}}$ Discretized function over subset $\mathcal{A}\subseteq\mathcal{I}$.
$\mathbf{f}$	$:= \mathbf{f}^{\mathrm{int}(\mathcal{I})}$ Discretized function over the domain.
$\mathbf{f}^n$	$:= \mathbf{f}^{\mathcal{T}^n}$ Discretized function over a traversal level.

6.5. Finite Difference.

ς^j	: Sub atomic finite difference operator.
σ^j	: Atomic finite difference operator that approximates $\mathcal{D}_{\mathbf{r}_{i,j}}$ locally.
σ_α	: An molecular finite difference operator
$\sigma_{(\cdot)q}$	: A not explicitly defined finite difference operator that approximates ∂_q
$\sigma_{(\cdot)\mathbf{v}}$	: A not explicitly defined finite difference operator that approximates $\mathcal{D}_{\mathbf{v}}$
σ_{Fq_k}	: The forward finite difference operator that approximates ∂_{q_k}
σ_{Bq_k}	: The backward finite difference operator that approximates ∂_{q_k}
$\sigma_{\Theta(\theta)q_k}$	: The θ finite difference operator that approximates ∂_{q_k}
σ_{Cq_k}	: The central finite difference operator that approximates ∂_{q_k}
S	: Finite Difference operator. General molecule.

6.6. Matrix representation.

$\boxed{\rightarrow}$	: The horizontal stack operator
$\boxed{\downarrow}$	: The vertical stack operator

$\boxed{\Rightarrow}$: The horizontal append operator

$\boxed{\Downarrow}$: The vertical append operator

$\mathrm{diag}(\mathbf{v})$: The matrix with the values of $\mathbf{v}$ on its diagonal, zero elsewhere.

$\delta_{i,j}$: Kronecker delta.

$\otimes$: Kronecker product

I_m : The $m \times m$ identity matrix.

J_m : The $m \times m$ offset matrix.

J_m^k : The $m \times m$ k-offset matrix.

∂J_m : The $m \times m$ boundary offset matrix.

∂J_m^k : The $m \times m$ boundary k-offset matrix.

$\mathbf{e}_k$: The k-th base vector.

$\mathbf{0}_{m_1,m_2}$: The $m_1 \times m_2$ zero matrix.

$\mathbf{1}_{m_1,m_2}$: The $m_1 \times m_2$ one matrix.

$\mathbf{0}_m$: The m dimensional zero vector.

$\mathbf{1}_m$: The m dimensional one vector.

$\lambda(A)$: Eigenvalue spectrum matrix A.

6.7. Creating the Linear System.

$Q^j_{\mathcal{A},\mathcal{B}}$: Approximation term matrix.

$M^S_{\mathcal{A},\mathcal{B}}$: Approximation term matrix.

A^n : Left hand side matrix

B^n : Right hand side matrix

$\mathbf{b}^n$: Right hand side vector.

6.8. r-refinement.

ξ $:= (\xi_1, \ldots, \xi_d)$ Pre adaptive uniform spatial coordinates.

θ : Pre adaptive uniform emporal coordinate.

$\lightning$ $:= (\lightning_1, \ldots, \lightning_{d_\mathbf{q}}) := (\xi_1, \ldots, \xi_d, \theta)$ Generalized pre adaptive uniform spatial temporal coordinates.

$\mathcal{J}$: Jacobian matrix of the adaptivity transformation.

$\mathbf{J}$: Discretized Jacobian matrix.