

A Test Set for an Adaptive Moving Grid PDE Solver with Time-Dependent Adaptivity Parameter

M. N. Zwarts

December 6, 2007

Abstract

We have tested a moving grid method with time-dependent adaptivity parameter on a wide variety of test problems. The real-time calculated adaptivity parameter makes the method more robust, removing the need to re-scale problems to fit the solver. When compared to the best results for a fixed adaptivity parameter in our computational efficiency measurements, this increased robustness more than compensates for the small decrease in performance. Finally, tests on a small subset of our test problems show that we can further improve results for models whose solution components differed by several orders of magnitude by calculating separate adaptivity parameters for each component.

Contents

1	Introduction	3
2	Moving Grid Method	4
2.1	Spatial discretization	4
2.2	Monitor function	7
2.3	Grid smoothing	9
2.4	Error tolerance	9
3	Test Problems	11
3.1	An Implosion Model	11
3.1.1	Error measurements	12
3.2	The Viscous Burger's Equation	16
3.2.1	Example I	17
3.2.2	Example II	19
3.3	A Shifting Pulse Model	20
3.4	A Perturbation Problem from Electro-Dynamics	22
3.5	An Advection-Reaction Model	24
3.6	A Flame Propagation Model	26
3.7	A Problem from Combustion	28
3.8	The Brusselator	30
3.9	The FitzHugh-Nagumo Equations	32
3.10	The Gray-Scott Reaction-Diffusion System	34
3.11	A Cylindrical Reaction-Diffusion PDE	37
3.12	Heat Flow of Harmonic Maps from Surfaces	39
3.13	A Basic Test Problem in Spherical Coordinates	41
3.14	The Implosion Model in Spherical Coordinates	43
3.15	A Brine Transport Model	45
3.15.1	Example I	46
3.15.2	Example II	47
3.15.3	Example III	48
3.16	A Model Describing Tumour Angiogenesis	49
3.16.1	Example I	50
3.16.2	Example II	51
3.17	A Gas Discharge Model	52
3.18	A Problem from Hydrology	56
3.19	A Model Describing the Formation of Liesegang Patterns	59
4	Possible Further Improvement	61
4.1	Three Implosions at Different Locations	62
4.2	Two Implosions at Different Locations and a Straight Line	65
4.3	Gas Discharge with new monitor function	67
5	Conclusions	68
A	Appendix: The Code	69
A.1	Overview	69
A.2	Updates	69

1 Introduction

A wide variety of phenomena from many different branches of science can be modelled with systems of partial differential equations (PDEs). When numerically solving PDEs, improving accuracy generally comes at the cost of increased computation times. Adaptive grid methods were developed to minimize this cost by dynamically improving accuracy in those parts of the domain where this is necessary, while allowing for a lower accuracy where possible.

There are two main types of adaptive grid methods: static-regridding and dynamic-regridding. In static-regridding methods (also known as h-refinement) gridpoints are added or removed as necessary, while in dynamic-regridding methods (also known as moving grid methods or r-refinement) a fixed number of gridpoints move continuously in space. Each type of refinement has its own advantages and disadvantages. Both types add a certain level of new complexity to the problem: for h-refinement it is necessary to have some error estimate or monitor to determine where points need to be added and where they can be removed, while r-refinement typically introduces an additional equation to prescribe the grid movement, based on intermediate solutions.

The method we will test is based on earlier work using the r-refinement approach [5]. The additional equation that prescribes the grid movement is based on the equidistribution principle, where a relevant quantity (a weight function) is distributed equally. A disadvantage of the weight function used in early moving grid methods, such as presented in [5], is that it requires the end user to choose an important parameter, known as the adaptivity parameter, in advance. To be able to choose a suitable adaptivity parameter, end users need to have a good understanding of both the moving grid method and the PDE system they wish to solve as well as some advance knowledge of its solution.

We will introduce a time-dependent adaptivity parameter, dynamically updated by the solver based on intermediate solutions. We hope that this will make the method more robust and user-friendly. Furthermore, if we encounter a model whose solution changes several orders of magnitude over the investigated timespan, a fixed adaptivity parameter that would be appropriate to the initial solution, might be too large or too small at some other point in time. This could result in a grid that is either too uniform (lacking accuracy in the most important area), or too focussed (lacking accuracy elsewhere). With a time-dependent adaptivity parameter, we expect to avoid such problems.

We test this moving grid method with time-dependent adaptivity parameter on a large set of test problems. Our purpose is twofold. Firstly, we mean to show that this method is well-suited to solving a large variety of different problems and secondly, we hope to provide a collection of possible test problems that can be used to test further improved methods.

In Chapter 2, we will discuss some of the theory behind the moving grid method. Topics range from the basics of the underlying numerical PDE solver, including the spatial discretization method and the class of PDEs we can solve, to the more specific choices for the monitor function and the time-dependent adaptivity parameter. Results are shown in Chapter 3, which includes a brief discussion of each particular test problem. Chapter 4 shows a possible weakness in the moving grid method with time-dependent adaptivity parameter as discussed in the previous chapters, as well as a possible solution. Finally, our conclusions can be found in Chapter 5.

2 Moving Grid Method

In this report we deal with a solver for systems of one-dimensional time-dependent PDEs with N_{PDE} component equations of the following form:

$$\frac{\partial u^j}{\partial t} = f^j(u), \quad x \in [x_L, x_R], \quad t \in [0, T_e], \quad j \in \{1, 2, \dots, N_{PDE}\}, \quad (1)$$

where $u^j = u^j(x, t)$ are components of the solution vector $u = (u^1, \dots, u^{N_{PDE}})^T$ and $f^j(u) = f^j(u, u_x, u_{xx}, x, t)$ with $j \in \{1, 2, \dots, N_{PDE}\}$, where u_x and u_{xx} denote the first and second derivatives of u with respect to x , respectively.

We investigate an adaptation of the moving grid method discussed in [5]. This method is based on a method-of-lines (MOL) approach, in which a system of form (1) is first converted into a system of ordinary differential equations (ODEs) by spatial discretization of the $f^j(u)$. Subsequently, the resulting semidiscrete system is solved through numerical integration in time, for which we use the public-domain code DASSL [2]. The class of PDEs that can be solved by this code is defined by:

$$\sum_{k=1}^{N_{PDE}} C_{j,k}(x, t, u, u_x) \frac{\partial u^k}{\partial t} = x^{-m} \frac{\partial}{\partial x} (x^m R_j(x, t, u, u_x)) - Q_j(x, t, u, u_x), \quad (2)$$

for $x \in [x_L, x_R]$, $t \in [t_0, T_e]$, $j \in \{1, 2, \dots, N_{PDE}\}$, $m \in \{0, 1, 2\}$, where $u \equiv (u^1, u^2, \dots, u^{N_{PDE}})^T$ is the solution vector and R_j and Q_j can sometimes be thought of as flux and sink terms, respectively [5]. For problems in cylindrical ($m = 1$) or spherical ($m = 2$) coordinates, $x_L \geq 0$. The problem of $x_L = 0$ in these cases, as well as the solution, is discussed in [8].

2.1 Spatial discretization

In our investigations, we focus on the first step: the spatial discretization of $f^j(u)$ onto a moving grid. The grid itself is a discretization of the physical domain $\Omega_p \equiv [x_L, x_R] \subset \mathbb{R}$. It consists of N time-dependent gridpoints $X_i(t)$, where $i \in \{1, 2, \dots, N\} \subset \mathbb{N}$:

$$x_L = X_0 < \dots < X_i(t) < X_{i+1}(t) < \dots < X_{N+1} = x_R. \quad (3)$$

For a uniform grid, X_i is given by:

$$X_i = x_L + i \cdot \frac{x_R - x_L}{N}, \quad i \in \{1, 2, \dots, N\}. \quad (4)$$

Equation (4) is basically a simple one-to-one map between i and X_i . We introduce a fixed computational domain $\Omega_c \equiv [0, 1] \subset \mathbb{R}$ with coordinate ξ , to discuss more general grid maps:

$$x = x(\xi, \theta), \quad \xi \in \Omega_c, \quad (5)$$

where $\theta = t$. To express the PDE system (1) in terms of these new coordinates (ξ, θ) , we use the identities:

$$\frac{\partial u}{\partial \xi} = \frac{\partial u}{\partial x} \frac{\partial x}{\partial \xi}, \quad (6)$$

and

$$\frac{\partial u}{\partial \theta} = \frac{\partial u}{\partial t} \frac{\partial t}{\partial \theta} + \frac{\partial u}{\partial x} \frac{\partial x}{\partial \theta} = \frac{\partial u}{\partial t} + \frac{\frac{\partial u}{\partial \xi}}{\frac{\partial x}{\partial \xi}} \frac{\partial x}{\partial \theta}, \quad (7)$$

where $u = u(x(\xi, \theta), t(\theta))$.

This results in a transformed PDE system of the following form:

$$\frac{\partial u}{\partial \theta} - \frac{\frac{\partial u}{\partial \xi}}{\frac{\partial x}{\partial \xi}} \frac{\partial x}{\partial \theta} = f(u). \quad (8)$$

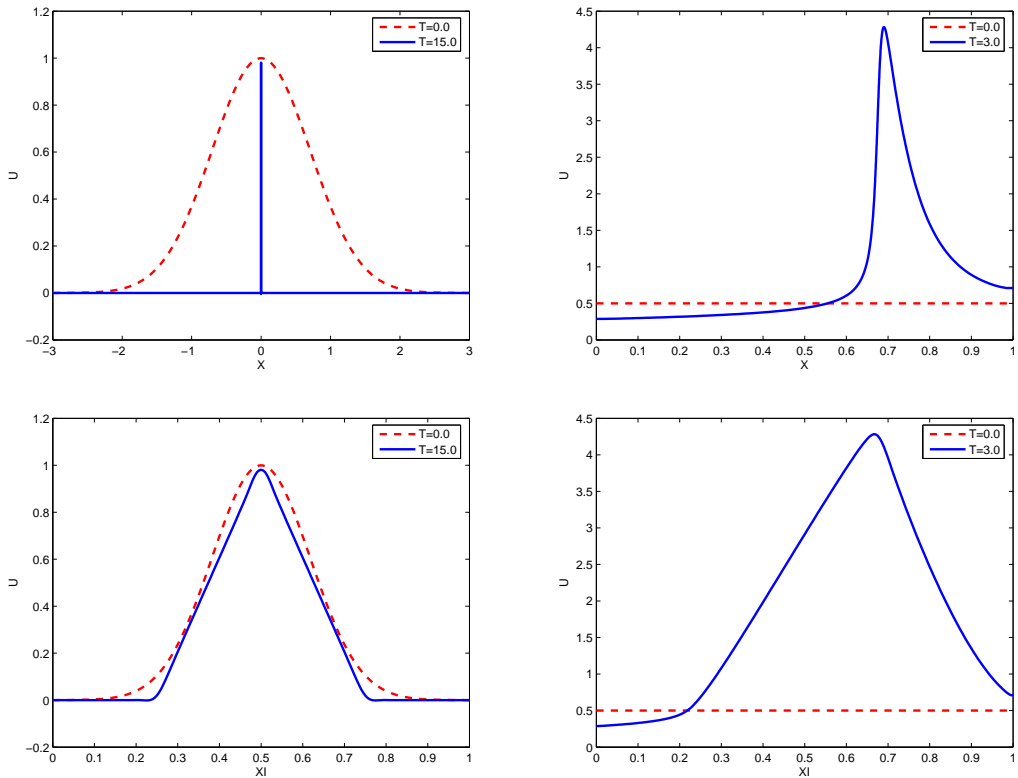


Figure 1: Approximate solution $U_i(t)$ as a function of X_i (top) and as a function of ξ_i (bottom) for two example problems: 'Implosion' (left, see section 3.1) and Brusselator (right, see section 3.8) at different times. $N = 201$.

To illustrate the difference that a remapping of the spatial coordinate can make, we show the numerical solutions U_i for two example problems from chapter 3 in figure 1, both as a function of X_i and as a function of ξ_i . The remapped functions $U_i(\xi)$ are clearly much smoother than the original functions $U_i(x)$. Such a remapping can be advantageous, since smoother functions generally require less gridpoints and, consequently, less computation time to be accurately approximated. The aim of a moving grid method is to find a dynamically updated map, such that for every step of the calculations, the function can be approximated accurately with a low number of gridpoints.

Following the variational approach used in [1], we will try to find the most appropriate map for the moving grid by minimizing a grid energy function $\mathcal{E}(\xi)$. Consider the following grid energy:

$$\mathcal{E}(\xi) = \frac{1}{2} \int_{\Omega_p} \left(\frac{\partial \xi}{\partial x} \right)^2 \frac{1}{M} dx, \quad (9)$$

where M is a monitor function. This choice of $\mathcal{E}(\xi)$ is minimized by the following Euler-Lagrange equation:

$$\frac{\partial}{\partial x} \left(\frac{1}{M} \frac{\partial \xi}{\partial x} \right) = 0. \quad (10)$$

The equivalent statement $M \frac{\partial x}{\partial \xi} = \text{constant}$, i.e.:

$$\frac{\partial}{\partial \xi} \left(M \frac{\partial x}{\partial \xi} \right) = 0, \quad (11)$$

is the equidistribution principle in 1D, which we will now present in discretized form. We replace the partial differential operators with the forward difference operator Δ^+ , defined by $\Delta^+ F_i = F_{i+1} - F_i$:

$$\frac{1}{\Delta^+ \xi_i} \Delta^+ \left(M_i(t) \frac{\Delta^+ X_i(t)}{\Delta^+ \xi_i} \right) = 0, \quad (12)$$

where $M_i(t)$ is the semidiscrete monitor function. Since ξ_i for $i \in \{1, 2, \dots, N\}$ forms a uniform grid on the computational domain Ω_c , we know that $\Delta^+ \xi_i = \frac{1}{N-1}$ is constant. Using the definition of the forward difference operator, we obtain the equidistribution principle in the following form:

$$\frac{M_{i+1}(t) \Delta^+ X_{i+1}(t) - M_i(t) \Delta^+ X_i(t)}{(N-1)^2} = 0, \quad (13)$$

which is equivalent to:

$$M_i(t) \Delta^+ X_i(t) = M_{i+1} \Delta^+ X_{i+1}(t), \quad i \in \{1, 2, \dots, N\}. \quad (14)$$

The semidiscrete approximation of the solution consists of N values $U_i(t)$: one for each $X_i(t)$ with $i \in \{1, 2, \dots, N\}$. Using a second order discretization for the spatial derivatives, we obtain the following semidiscrete version of (8):

$$\frac{dU_i^j}{dt} - \frac{U_{i+1}^j - U_{i-1}^j}{X_{i+1} - X_{i-1}} \frac{dX_i}{dt} = F_i^j, \quad i \in \{1, 2, \dots, N\}, \quad t \in [0, T_e], \quad (15)$$

where F_i^j is the semidiscrete approximation of $f^j(u)$ from (1). The system described by (15) consists of N ODEs with $2N$ unknowns, so we will need N additional equations in order to find a solution. The semidiscrete form of the equidistribution principle (14), provides exactly what we need. Introducing the point concentration $n_i = (\Delta X_i)^{-1}$, we rewrite (14) as:

$$\frac{n_i}{n_{i-1}} = \frac{M_i}{M_{i-1}}, \quad i \in \{1, 2, \dots, N\}. \quad (16)$$

The complete ODE system formed by (15) and (16) is the final result of the spatial discretization MOL step. If combined with appropriate initial conditions

and boundary values, this system can be solved by numerical integration in time. Clearly, (16) implies that the relative point concentration at every point in the moving grid is completely determined by the relative value of the solution-dependent monitor function. The challenge for creating a successful moving grid method then lies in finding the most suitable monitor function.

2.2 Monitor function

The aim of a moving grid method is to have a higher point concentration in those parts of the domain where higher accuracy is needed. We will refer to those parts as 'critical areas'. Generally, what makes an area critical depends on the type of problem at hand. For our purposes, we will consider a region critical when the solution u changes rapidly in space and non-critical when u is more or less constant.

The most obvious measure for the spatial activity of u is the derivative $\frac{\partial u}{\partial x}$. In some cases, higher order derivatives may be preferable. For example, a part of a function where the derivative is very large, but constant, can be accurately described with only two points, while a part that resembles a sinusoid function $u(x) = \sin(x)$ has a derivative that is bounded by 1, but clearly needs more than two points for an accurate description. Nonetheless, moving grid methods based on the first derivative have worked well in previous experiments (e.g. [5]) and we will not seek improvements in the line of higher order derivatives.

Based on the previous discussion, M_i will be a function of the local partial derivative:

$$M_i \sim \left| \frac{\Delta U_i}{\Delta X_i} \right|, i \in \{1, 2, \dots, N\}, \quad (17)$$

with $\Delta U_i \equiv U_{i+1} - U_i$ and $\Delta X_i \equiv X_{i+1} - X_i$.

The monitor function should be positive everywhere. While taking the absolute value guarantees non-negativity, the partial derivative can become zero. To keep the monitor function away from zero, an additional factor, α , is necessary in the definition of the monitor function:

$$M_i = \alpha + \sum_{j=1}^{N_{PDE}} \left| \frac{\Delta U_i^j}{\Delta X_i} \right|, i \in \{1, 2, \dots, N\}. \quad (18)$$

The choice of this α , which we will call the adaptivity parameter, is a major factor in determining what the moving grid will look like. If α is much larger than the derivative, then the monitor function and, consequently, the gridpoint concentration will be nearly constant, while a very small α may result in too low gridpoint concentrations in areas where the derivative is small. Finding the 'correct', or at least 'near-optimal', value of α can be a difficult task. For example, a problem from astronomy and a problem from chemistry will work on scales that vary in many orders of magnitude, making it unlikely that a single value of α would work well for both. While it should generally be possible to rescale most problems such that a single choice of α (e.g. $\alpha = 1$) works for all, this rescaling procedure quickly adds up to a lot of work, which we hope to avoid.

In past experiments [5], [9], the choice of α had to be made in advance by the user, while more recently [1], there have also been experiments where the adaptivity parameter was calculated in real time. We investigate such a moving

grid method with time-dependent adaptivity parameter $\alpha(t)$. In many cases, the monitor function M_i was a more complicated function of the spatial derivative. For example, the arclength monitor used in [5]:

$$M_i = \sqrt{\alpha + \sum_{j=1}^{N_{PDE}} \left| \frac{\Delta U_i^j}{\Delta X_i} \right|^2}, i \in \{1, 2, \dots, N\}. \quad (19)$$

Apart from making α time-dependent, we've decided to keep the very basic monitor function (18). The $\alpha(t)$ we've chosen is basically a weighted average of the derivative-term $\sum_{j=1}^{N_{PDE}} \left| \frac{\Delta U_i^j}{\Delta X_i} \right|$ in (18):

$$\alpha(t) = \left\langle \sum_{j=1}^{N_{PDE}} \left| \frac{\Delta U_i^j}{\Delta X_i} \right| \right\rangle = \frac{\sum_{i=1}^N \sum_{j=1}^{N_{PDE}} \left| \frac{\Delta U_i^j}{\Delta X_i} \right| \Delta X_i}{\sum_{i=1}^N \Delta X_i} \simeq \frac{\sum_{i=1}^N \sum_{j=1}^{N_{PDE}} \left| \Delta U_i^j \right|}{x_R - x_L}. \quad (20)$$

Finally, we make one smaller change to the monitor function: to control the ratio between the number of points in the critical areas and the number of points in non-critical areas, we introduce a coefficient $\beta \in (0, 1)$:

$$M_i = (1 - \beta)\alpha(t) + \beta \sum_{j=1}^{N_{PDE}} \left| \frac{\Delta U_i^j}{\Delta X_i} \right|, i \in \{1, 2, \dots, N\}. \quad (21)$$

For brevity of notation, we will consider a single-component PDE (i.e. $N_{PDE} = 1$) in the following discussion of the role played by this coefficient β . Following [1], we divide the average of the critical part of the monitor function by the average of the complete monitor function to estimate the ratio R_C of points in critical areas:

$$R_C = \frac{\langle \beta \left| \frac{\partial u}{\partial x} \right| \rangle}{\langle M \rangle} = \frac{\int_{x_L}^{x_R} \beta \left| \frac{\partial u}{\partial x} \right| dx}{\int_{x_L}^{x_R} M dx} \approx \frac{\sum_{i=1}^N \beta \left| \frac{\Delta U_i}{\Delta X_i} \right| \Delta X_i}{\sum_{i=1}^N M_i \Delta X_i}. \quad (22)$$

Note that, since they cancel, we've left out a factor $\frac{1}{x_R - x_L}$ from both numerator and denominator in (22). However, reintroducing this factor, we see that the numerator:

$$\sum_{i=1}^N \beta \left| \frac{\Delta U_i}{\Delta X_i} \right| \frac{\Delta X_i}{x_R - x_L} = \beta \alpha, \quad (23)$$

and that the denominator:

$$\sum_{i=1}^N M_i \frac{\Delta X_i}{x_R - x_L} = \sum_{i=1}^N (1 - \beta)\alpha(t) + \beta \left| \frac{\Delta U_i}{\Delta X_i} \right| \frac{\Delta X_i}{x_R - x_L} = \alpha - \beta\alpha + \beta\alpha = \alpha, \quad (24)$$

from which we conclude that $R_C = \frac{\beta\alpha}{\alpha} = \beta$. Consequently, the value $\beta = 0.8$, which we've used in all our experiments should result in a grid with approximately 80% of its gridpoints in the critical areas.

2.3 Grid smoothing

The ODE system (15), (16) with monitor function (21) prescribes the movement of the gridpoints $X_i(t)$ as a function of solution $U_i(t)$. Since we generally don't know the solution's behaviour in advance, this makes the grid movement somewhat unpredictable. Considering the fact that moving grid methods are designed specifically to solve problems with extreme spatial and temporal variations, we may expect similarly large variations in the point concentration. Spatial accuracy may actually decrease, if the point concentration varies too rapidly in space. In the temporal domain, there is a risk of grid oscillations, which may hinder the numerical integration steps.

There are different ways to obtain a smoother grid and hopefully prevent such problems. For example, in [1], a smoothed monitor function is used. However, we choose to use the approach from [5] and introduce two smoothing parameters τ and κ into equation (16):

$$\frac{\tilde{n}_{i-1} + \tau \dot{\tilde{n}}_{i-1}}{M_{i-1}} = \frac{\tilde{n}_i + \tau \dot{\tilde{n}}_i}{M_i}, i \in \{1, 2, \dots, N\}. \quad (25)$$

While the temporal smoothing parameter τ is explicit in (25), the spatial smoothing parameter κ is introduced implicitly with the smoother point concentration function \tilde{n}_i :

$$\tilde{n}_i = n_i - \kappa(\kappa + 1)(n_{i+1} - 2n_i + n_{i-1}), i \in \{1, 2, \dots, N\}, \quad (26)$$

which puts a bound on the variation in ΔX_i :

$$\frac{\kappa}{\kappa + 1} \leq \frac{n_{i-1}}{n_i} \leq \frac{\kappa + 1}{\kappa}, i \in \{1, 2, \dots, N\}. \quad (27)$$

Note that the extreme values $\kappa = \infty$ and $\tau = \infty$ result in a uniform grid, while the other extremes $\kappa = 0$ and $\tau = 0$ simply give the equations without smoothing. Since we focus on the role of α , we have tried to vary κ and τ as little as possible. We've used the values $\kappa = 2$ and $\tau = 10^{-3}$ in all experiments described in the following chapters, except where we explicitly state otherwise.

Using a fixed value for the temporal smoothing parameter has the disadvantage of making the solver less robust: given a PDE system whose solution changes on a timescale much smaller than τ , the grid will not adapt quickly enough to keep up with the changes in the solution. A possible solution is to set $\tau = 10^{-2}T_{crit}$, where T_{crit} is a user-defined parameter that denotes the critical timescale on which the solution changes. If the user has insufficient prior knowledge of the solution to determine T_{crit} , a good rule of thumb is to use $\tau = 10^{-3}T_e$.

2.4 Error tolerance

The final step of the moving grid method is to solve the semidiscrete system consisting of $N_{PDE} + 1$ equations (i.e. N_{PDE} model equations + 1 moving grid equation) through numerical integration. We use the public-domain DASSL code (see [2]). One DASSL parameter that we've not discussed so far, but which can greatly affect performance is the time-tolerance, or error tolerance.

We quote comments in the DASSL code: "You must assign relative (RTOL) and absolute (ATOL) error tolerances to tell the code how accurately you want the solution to be computed. The tolerances are used by the code in a local error test at each step which requires roughly that $\text{ABS}(\text{LOCAL ERROR}) \leq \text{RTOL} * \text{ABS}(Y) + \text{ATOL}$ for each vector component." Smaller values for the error tolerance will usually give a smaller error in the final numerical solution, but increase the required computation time. We use a value of 10^{-6} for both the relative and the absolute error tolerance in all our runs.

3 Test Problems

In this chapter, we show the results obtained with the moving grid method with time-dependent adaptivity parameter for a large number of test problems. We start with a simple model that highlights the strength of the moving grid method (3.1), followed by several well-known problems from literature, roughly in order of increasing complexity (3.2-3.10). Next we look at some examples in cylindrical (3.11 and 3.12) and spherical coordinates (3.13 and 3.14), saving the most complex examples in Cartesian coordinates for last (3.15-3.19).

We have chosen to use a default number of gridpoints $N = 101$, with lower values for problems with very smooth solutions (e.g. section 3.13) and higher values for problems whose solutions have a large number of critical areas (e.g. section 3.10). For problems with multiple component solutions (i.e. $N_{PDE} > 1$), we generally use all N_{PDE} components to calculate the monitor function. Only in two cases (sections 3.18 and 3.19) where one of the components is actually an artificial variable, introduced to enable us to solve a higher order PDE, do we make an exception: we exclude the artificial variable from the monitor function.

3.1 An Implosion Model

Our first example is a simple PDE with a known exact solution $u_{ex}(x, t) = e^{-(e^t x)^2}$. This exact solution is a Gaussian function with constant height whose width decreases with time (t). It is a useful test problem because it has a clear critical area and the height difference between the exact and numerical solutions provides a simple, yet effective error measure.

A two-dimensional "implosion" PDE can be found in [3]. We use the following one-dimensional version (for $x \in [-3, 3]$ and $t \in [0, T_e]$):

$$\frac{\partial u}{\partial t} = x \frac{\partial u}{\partial x}, \quad (28)$$

with initial and boundary conditions:

$$u(x, 0) = e^{-x^2}, \quad u(-3, t) = e^{-(3e^t)^2}, \quad u(3, t) = e^{-(3e^t)^2}. \quad (29)$$

Figure 2 shows that the moving-grid method with time-dependent α manages to solve the problem quite well with 401 gridpoints. Even at $T_e = 20.0$, where the width of the Gaussian is of order $O(10^{-8})$, the error in its maximum value is only 1.05%. If we were to use a fixed, uniform grid, we'd expect to need at least 10^8 points, just to be able to show such a narrow Gaussian, let alone obtain it as a numerical solution.

The adaptivity parameter $\alpha(t)$ for this PDE has an almost constant value of approximately 0.33. Remember that $\alpha(t) = \frac{\int_{x_L}^{x_R} |\frac{\partial u}{\partial x}| dx}{x_R - x_L}$ and we calculate it as follows:

$$\alpha(t) = \frac{\int_{x_L}^{x_R} |\frac{\partial u}{\partial x}| dx}{x_R - x_L} \approx \frac{\sum_{i=1}^{N-1} |\frac{\Delta u_i}{\Delta x_i}| \Delta x_i}{x_N - x_1} = \frac{\sum_{i=1}^{N-1} |\Delta u_i|}{x_N - x_1}, \quad (30)$$

where $\Delta x_i = x_{i+1} - x_i$ and $\Delta u_i = u_{i+1} - u_i$.

Now, note that our solution $u(x, t) \approx u_{ex}(x, t) = e^{-(e^t x)^2}$ can be split up in two halves: the left half increases monotonically from ≈ 0 to ≈ 1 , while the right half decreases monotonically from ≈ 1 to ≈ 0 . This makes it easy to see that

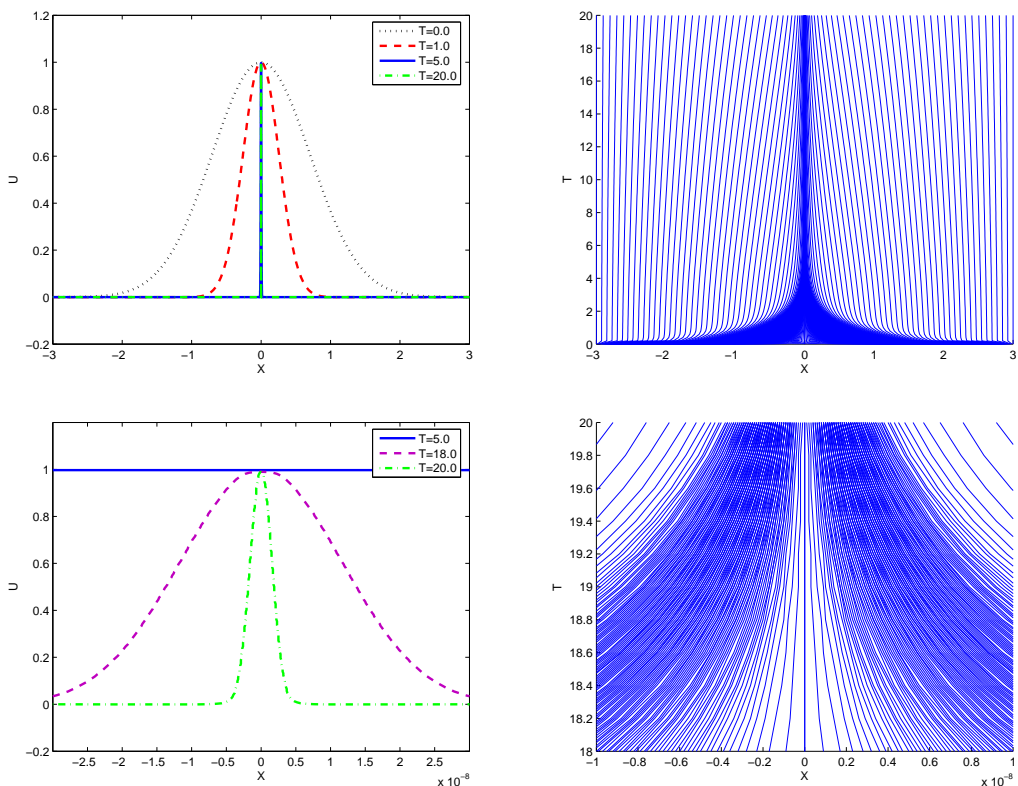


Figure 2: Results for the implosion problem: solution $u(x, t)$ (top left), grid movement $X_i(t)$ (top right) and zoomed in versions of both (bottom left and right). $N = 401$.

we can estimate $\sum_{i=1}^{N-1} |u_{i+1} - u_i|$ for each half at 1 and their sum at 2. Since we also know that $x_N - x_1 = x_R - x_L = 3 - (-3) = 6$, we find the following estimate for $\alpha(t) = 2/6 \approx 0.33$, which agrees with our numerical results.

3.1.1 Error measurements

We use the following, rather simple indicator of the error E in our results:

$$\begin{aligned}
 E &= |\max\{u_{ex}(x) \mid x \in [x_L, x_R]\} - \max\{u(x) \mid x \in [x_L, x_R]\}| \\
 &= |1 - \max\{u(x) \mid x \in [x_L, x_R]\}|.
 \end{aligned} \tag{31}$$

Note that this only applies to the "implosion" PDE and is only valid as long as the solution still has a Gaussian shape.

We performed error measurements for three different grid types: (I) a fixed grid, (II) a moving grid with constant adaptivity parameter $\alpha = \alpha_f$ and (III) a moving grid with time-dependent adaptivity parameter $\alpha(t)$. We used monitor function (21) for both moving grid methods with fixed and with time-dependent α . For each type, we performed several runs with different numbers of grid-points: $N = 101, N = 201, N = 401, N = 801, N = 1601$. For the moving grid

methods, both with fixed and with time-dependent α , we tested with four different values of T_e . We didn't test extensively with the fixed grid for $T_e > 5.0$, because preliminary tests showed that the error became too large, even with $N = 1601$.

Besides the error, we also measured the computing time T_C required for each run. All tests were performed on the same machine: a portable personal computer with a 1.6GHz Intel Centrino CPU and 512MB of RAM. Detailed results are given in tables 1-5. For some choices of α and N , we were unable to complete runs with $T_e = 20.0$. In some cases we encountered a "failure to converge" in the numerical integration, while other runs simply took so long that we had to manually break off the run due to time limitations. All such unsuccessful runs are denoted by "***" in the tables.

In figure 3, we've plotted the error E as a function of the required computation time T_C for our various runs, to directly compare the results of the different methods. Points in the lower left corner indicate better performance, while those in the upper right are worse. Unsuccessful runs ("***" in tables 1-5) are left out completely.

Clearly, the moving grid methods (II and III) perform much better than the fixed grid method (I). The performance differences between the different moving grid methods are smaller, but we can still conclude that method (III) only outperforms method (II) with $\alpha_f = 10.0$. However, we observed that the time-dependent $\alpha(t)$ was constant at approximately 0.33 for this problem. Consequently, the main difference between method (III) and method (II) with $\alpha_f = 0.1$ or $\alpha_f = 1.0$, is the additional computation time that method (III) spends to calculate $\alpha(t)$. Figure 3 indeed shows that, for equal numbers of points N , method (III) has a higher computation time T_C than method (II), but a very similar error E .

A major advantage of the moving grid method with time-dependent adaptivity parameter that is not reflected in figure 3, is its robustness. To use method (II) successfully for any given problem, a user needs to find an appropriate adaptivity parameter, either through trial and error, or by using prior knowledge of the solution. Method (III) effectively eliminates this requirement.

Table 1: Time-dependent $\alpha(t)$

N	$T_e = 5.0$		$T_e = 10.0$		$T_e = 15.0$		$T_e = 20.0$	
	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$
101	1.8	1.53	2.4	3.40	2.8	6.84	1069.0	41.29
201	3.2	0.65	4.3	1.27	4.8	1.91	10.0	2.61
401	5.5	0.30	7.5	0.57	8.8	0.81	128.0	1.05
801	8.9	0.14	23.2	0.27	47.2	0.38	>9000	***
1601	16.7	0.07	160.8	0.13	547.3	0.18	***	***

Table 2: Fixed $\alpha_f = 0.1$

N	$T_e = 5.0$		$T_e = 10.0$		$T_e = 15.0$		$T_e = 20.0$	
	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$
101	0.9	1.41	1.1	3.10	1.4	6.22	1.9	38.53
201	1.5	0.60	1.9	1.14	2.5	1.70	3.6	2.33
401	2.8	0.27	3.8	0.51	4.9	0.72	67.1	0.93
801	5.7	0.13	30.0	0.24	49.9	0.34	>3600	***
1601	16.9	0.06	211.4	0.11	326.9	0.16	***	***

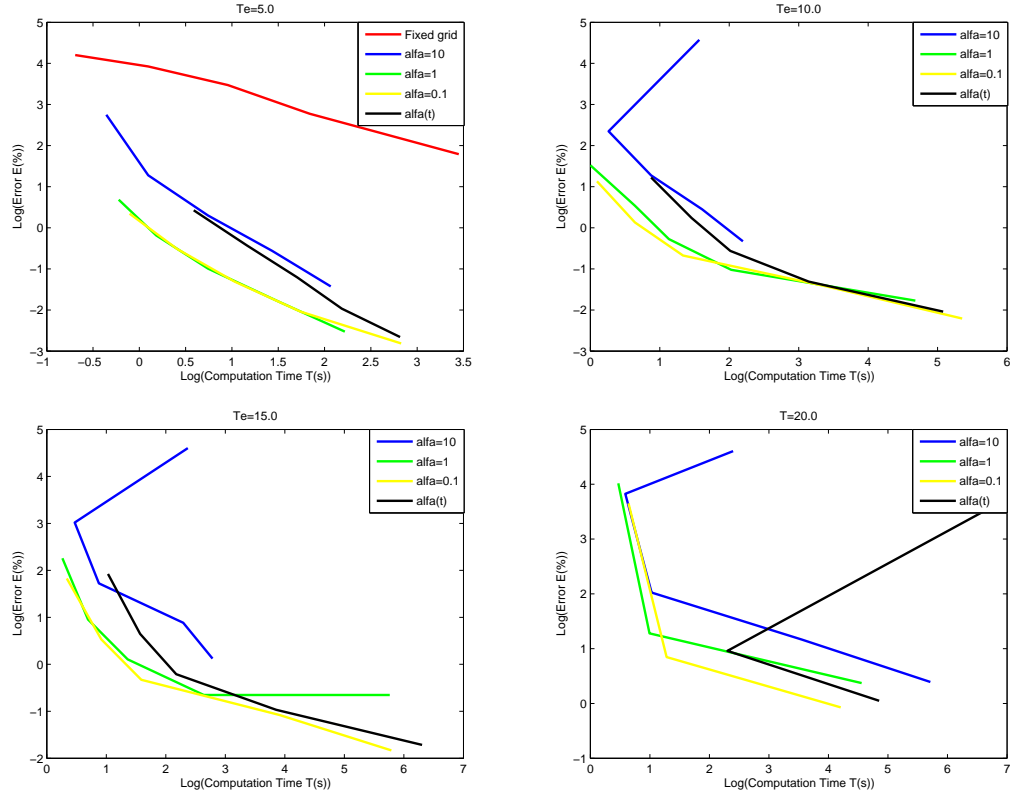


Figure 3: Computational efficiency diagram for the implosion problem. $T_e = 5.0$ (top left), $T_e = 10.0$ (top right), $T_e = 15.0$ (bottom left), $T_e = 20.0$ (bottom right). Each line connects the results of a single method (fixed grid (I), moving grid (II) with $\alpha_f = 0.1$, $\alpha_f = 1.0$, or $\alpha_f = 10.0$, or moving grid with time-dependent $\alpha(t)$ (III)) with different numbers of gridpoints: $N = 101$, $N = 201$, $N = 401$, $N = 801$, $N = 1601$.

Table 3: Fixed $\alpha_f = 1.0$

N	$T_e = 5.0$		$T_e = 10.0$		$T_e = 15.0$		$T_e = 20.0$	
	$T_C(\text{s})$	$E(\%)$	$T_C(\text{s})$	$E(\%)$	$T_C(\text{s})$	$E(\%)$	$T_C(\text{s})$	$E(\%)$
101	0.8	1.98	1.0	4.58	1.3	9.57	1.6	55.58
201	1.2	0.83	1.9	1.71	2.0	2.60	2.7	3.59
401	2.1	0.37	3.1	0.76	3.9	1.11	95.3	1.45
801	4.4	0.17	7.6	0.36	14.0	0.52	***	***
1601	9.2	0.08	107.7	0.17	318.0	0.52	***	***

Table 4: Fixed $\alpha_f = 10.0$

N	$T_e = 5.0$		$T_e = 10.0$		$T_e = 15.0$		$T_e = 20.0$	
	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$	$T_C(s)$	$E(\%)$
101	0.7	15.65	4.8	96.80	10.7	99.76	11.0	99.99
201	1.1	3.59	1.3	10.48	1.6	20.48	1.8	45.87
401	2.1	1.35	2.4	3.58	2.4	5.60	2.8	7.56
801	4.2	0.57	5.0	1.57	9.9	2.42	35.1	3.20
1601	7.9	0.24	9.0	0.72	16.2	1.13	302.7	1.48

Table 5: Fixed grid

$T_e = 5.0$		
N	$T_C(s)$	$E(\%)$
101	0.5	66.95
201	1.1	50.66
401	2.6	32.20
801	6.3	16.01
1601	31.5	6.00

3.2 The Viscous Burger's Equation

We also test the viscous Burger's equation with two different sets of initial and boundary conditions (for $x \in [0, 1]$ and $t \in [0, T_e]$ in both cases). Both examples are taken from [4]. The viscous Burger's equation:

$$\frac{\partial u}{\partial t} = \varepsilon \frac{\partial^2 u}{\partial x^2} - u \frac{\partial u}{\partial x}, \quad (32)$$

where $\varepsilon = 5.0 * 10^{-4}$ is a constant that determines the relative importance of the viscous term $\varepsilon \frac{\partial^2 u}{\partial x^2}$, as well as the steepness of the fronts in the initial condition for Example I.

3.2.1 Example I

In the first case, the initial and boundary conditions are all determined by a function $u_{ex}(x, t)$:

$$\begin{aligned} u(x, 0) &= u_{ex}(x, 0), \\ u(0, t) &= u_{ex}(0, t), \\ u(1, t) &= u_{ex}(1, t). \end{aligned} \quad (33)$$

We choose the following function $u_{ex}(x, t)$:

$$u_{ex}(x, t) = 1 - \frac{0.9r_1}{r_1 + r_2 + r_3} - \frac{0.5r_1}{r_1 + r_2 + r_3}, \quad (34)$$

where:

$$r_1 = e^{(-\frac{x-\frac{1}{2}}{20\varepsilon} - \frac{99t}{400\varepsilon})}, \quad r_2 = e^{(-\frac{x-\frac{1}{2}}{4\varepsilon} - \frac{3t}{16\varepsilon})}, \quad r_3 = e^{(-\frac{x-\frac{3}{8}}{2\varepsilon})}. \quad (35)$$

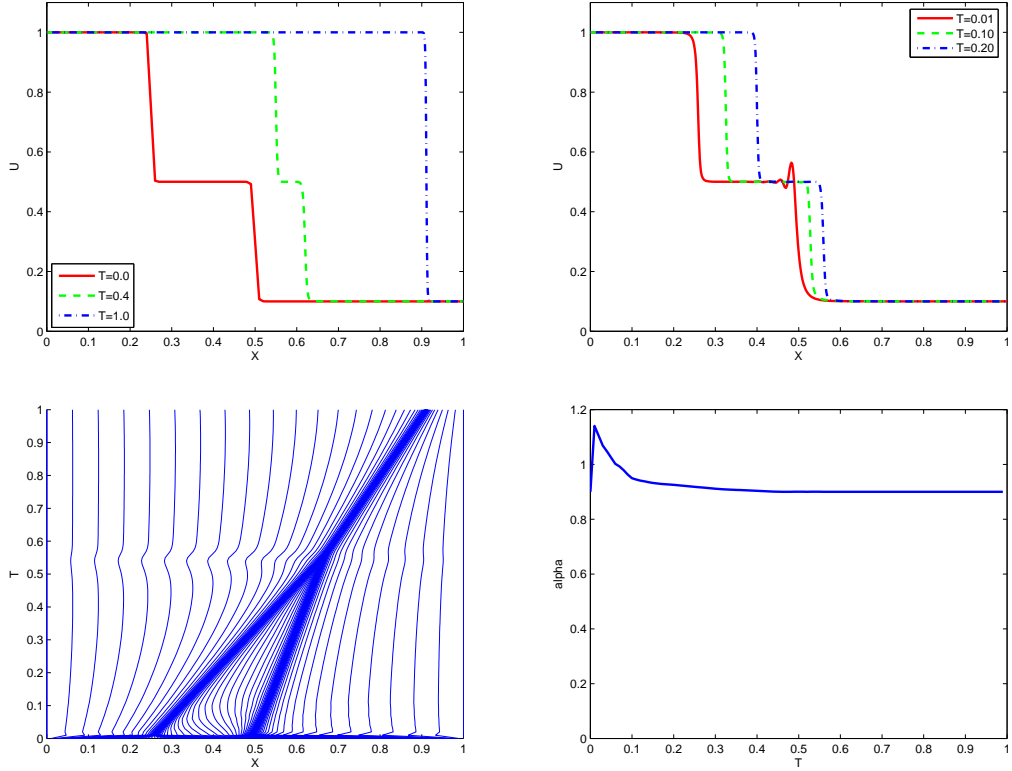


Figure 4: Results for the Burger's equation, example I: solution $u(x, t)$ at different times (top left and right), grid movement $x_i(t)$ (bottom left) and $\alpha(t)$ (bottom right). $N = 101$.

We show the results for 101 gridpoints in figure 4 and for 201 gridpoints in 5. The initial solution consists of a function that has two steep gradients, one around $x = 0.25$ and one around $x = 0.5$, while being constant between the two

fronts, as well as between each front and the nearest border. We see that the higher front on $u(x, t)$ moves to the right faster than the lower front. Eventually, the fast front overtakes the slower one and they "merge" into a single front with their combined height. The gridpoints follow the large gradient areas (first the two individual fronts and later the large, merged front). The grid movement plot nicely shows that the speed with which the merged front moves lies between the speeds of the individual fronts.

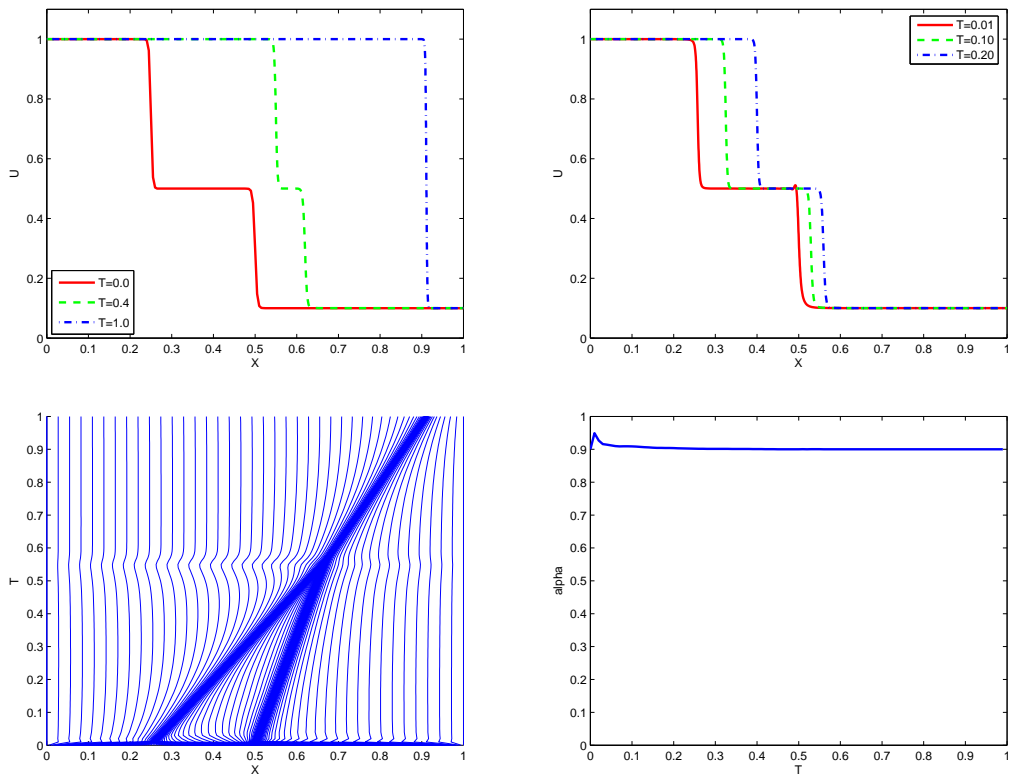


Figure 5: Results for the Burger's equation, example I: solution $u(x, t)$ at different times (top left and right), grid movement $x_i(t)$ (bottom left) and $\alpha(t)$ (bottom right). $N = 201$.

A numerical error in the form of an oscillating pattern on the right-hand side of the area between the two moving fronts is most visible at time $t = 0.01$ in figure 4. Interestingly, this error is reflected by a spike in the value of $\alpha(t)$, approximately between $t = 0$ and $t = 0.1$. The larger number of gridpoints used to obtain figure 5 results in a much smaller error, but we can still see a little spike in the value of $\alpha(t)$.

3.2.2 Example II

The second Viscous Burger's Equation test case has the same basic equations, but different initial and boundary conditions:

$$\begin{aligned} u(x, 0) &= \sin(2\pi x) + \frac{1}{2} \sin(\pi x), \\ u(0, t) &= u(1, t) = 0. \end{aligned} \quad (36)$$

This problem can also be found in [7].

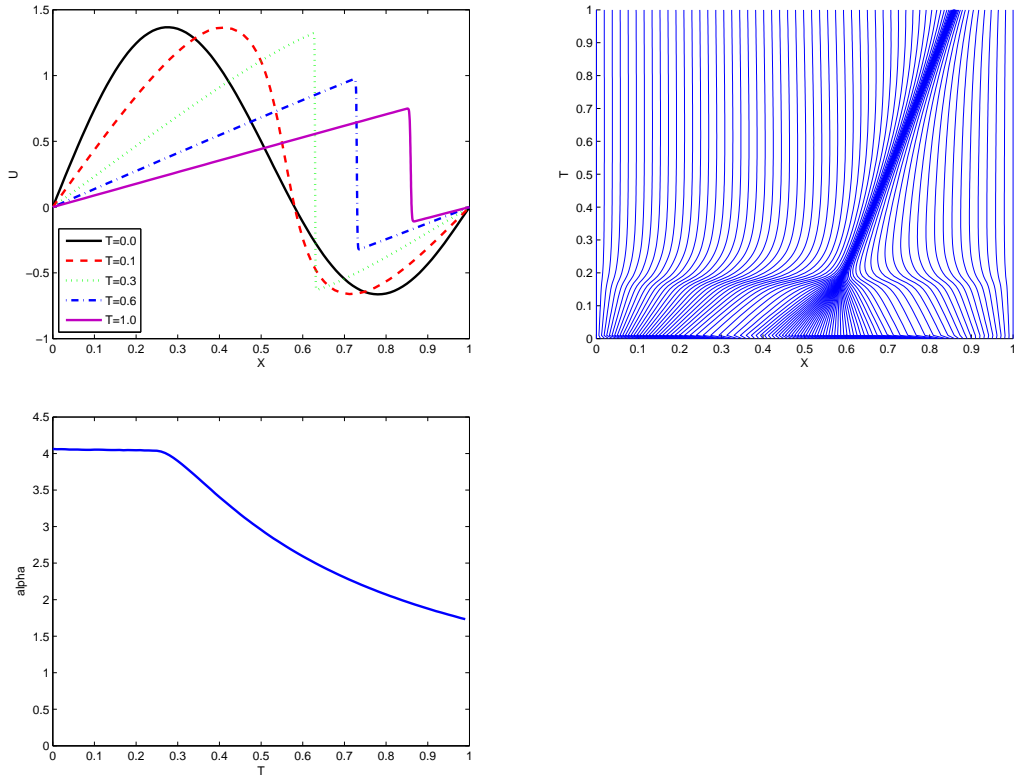


Figure 6: Results for the Burger's equation, example II: solution $u(x, t)$ (top left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 101$.

We see in figure 6, that the initial combination of sine-functions quickly evolves into a steep front moving to the right, somewhat similar to our observations in the previous Burger's equation example. The main difference is the fact that, in this case, the height of the travelling front decreases, whereas it was constant in the earlier example. Consequently, $\alpha(t)$ decreases as well. The gridpoint lines outside of the critical area in figure 6's grid movement plot approach straight, vertical lines as time t approaches T_e . We conclude that the gridpoint density outside of the critical area becomes nearly a constant. For a fixed α , we would expect the gridpoint density outside of the critical area to increase as the height of the travelling front decreases.

3.3 A Shifting Pulse Model

Similar to the Implosion problem (Section 3.1), the shifting pulse problem is a purely artificial PDE, designed with a specific exact solution in mind. In this case, the PDE is designed such that the exact solution $u_{ex}(x, t)$ is a Gaussian function (a pulse) that shifts back and forth along the x -axis while its height oscillates between 0 and 2:

$$u_{ex}(x, t) = e^{-A(x-r_B(t))^2}(1 - \sin(C\pi t)), \quad (37)$$

where $r_B(t) = \frac{1}{4}(2 + \sin(B\pi t))$. This PDE was used as a test case in [7] and we've chosen the same values for the constants: $A = 320$, $B = 1$ and $C = 2$. Different values could be chosen to change the width and steepness of the pulse (A), the period of the horizontal shifting (B), or the period of the vertical oscillation (C).

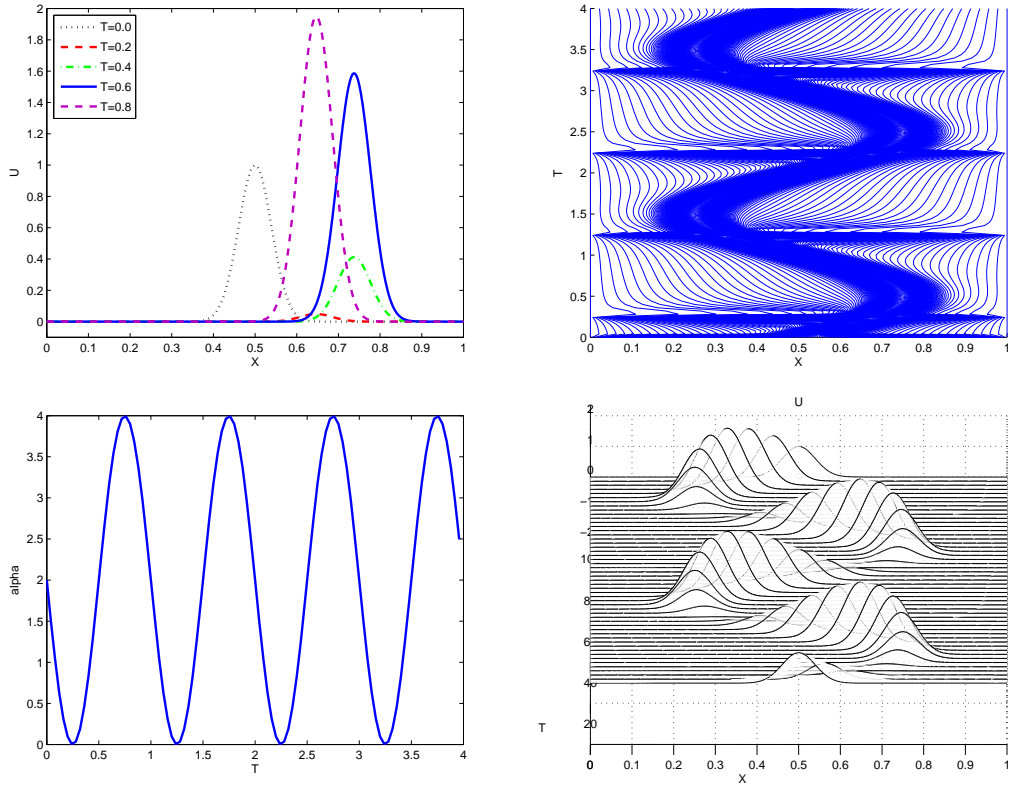


Figure 7: Results for the shifting pulse problem: solution $u(x, t)$ (top left), grid movement $x_i(t)$ (top right), $\alpha(t)$ (bottom left) and a waterfall plot of $u(x, t)$ (bottom right). $N = 201$.

A PDE with this exact solution can be obtained from the following basic PDE:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f(x, t), \quad (38)$$

simply by choosing $f(x, t) = \frac{\partial u_{ex}}{\partial t} - \frac{\partial^2 u_{ex}}{\partial x^2}$. The initial and boundary conditions ($x \in [0, 1]$) are also prescribed by $u_{ex}(x, t)$:

$$\begin{aligned} u(x, 0) &= u_{ex}(x, 0) = e^{-A(x-\frac{1}{2})^2}, \\ u(0, t) &= e^{-A(r_B(t))^2} (1 - \sin(C\pi t)), \\ u(1, t) &= e^{-A(1-r_B(t))^2} (1 - \sin(C\pi t)). \end{aligned} \tag{39}$$

It is difficult to see the oscillating behaviour of the shifting pulse in the $u(x)$ -plots in figure 7, so we've added a "waterfall" plot. This allows us to see that the solution $u(x, t)$ indeed behaves much like $u_{ex}(x, t)$: it shifts from left to right with period $2/B = 2$ and its height oscillates with period $C/2 = 1$.

The oscillating behaviour is clearly visible in both $\alpha(t)$ and the grid movement. However, we observe disturbances in the grid, whenever the pulse height becomes zero. When the pulse height is zero, the whole function is zero, as well as its derivative. Consequently, $\alpha(t)$ also becomes zero. Since one of the reasons to add α to the monitor function is to prevent it from becoming zero, we clearly have a problem when $\alpha(t)$ becomes zero. We've chosen to solve this problem by setting the whole monitor function to 1, resulting in a uniform grid, whenever $\alpha(t)$ becomes zero. The disturbances we see in the grid movement plot, are simply due to the grid going from non-uniform to uniform and back.

3.4 A Perturbation Problem from Electro-Dynamics

This example is a perturbation problem from electro-dynamics that we've taken from [5] (Example I). The PDE system is:

$$\begin{aligned}\frac{\partial u}{\partial t} &= \varepsilon p \frac{\partial^2 u}{\partial x^2} - g(u - v), \\ \frac{\partial v}{\partial t} &= p \frac{\partial^2 v}{\partial x^2} + g(u - v),\end{aligned}\quad (40)$$

where $\varepsilon = 0.143$, $p = 0.1743$ and $\eta = 17.19$ are constants and $g(z) = e^{\eta z/3} - e^{-2\eta z/3}$.

The initial and boundary conditions are:

$$\begin{aligned}u(x, 0) &= 1, & v(x, 0) &= 0, \\ \frac{\partial u}{\partial x}(0, t) &= 0, & v(0, t) &= 0, \\ u(1, t) &= 1, & \frac{\partial v}{\partial x}(1, t) &= 0.\end{aligned}\quad (41)$$

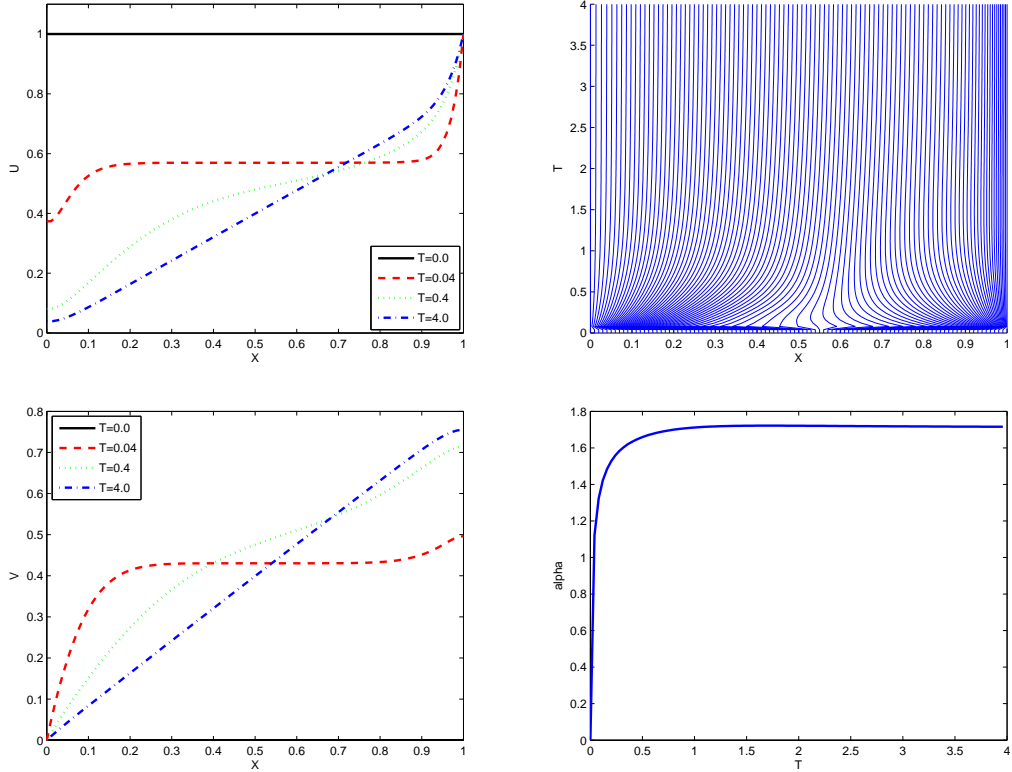


Figure 8: Results for the electro-dynamic perturbation problem: solution $u(x, t)$ (top left), grid movement $x_i(t)$ (top right), $v(x, t)$ (bottom left) and $\alpha(t)$ (bottom right). $N = 101$.

We see in figure 8 that, after starting as constant initial functions, $u(x, t)$ and $v(x, t)$ almost instantly generate very steep gradients at the boundaries and then slowly evolve into smoother profiles, reaching "nearly" steady state solutions

around $t = 1.5$. This behaviour is also visible in the grid movement. The grid points initially move towards the left and right, where the steep boundary layers appear. Then, they slowly move back towards an almost uniform distribution, except near $x = 1.0$, where the grid point density remains relatively high. While $\alpha(t)$ starts at 0, it quickly increases when the steep boundary layers form and it reaches its constant value of 1.7 before $u(x, t)$ and $v(x, t)$ reach their steady state profiles.

3.5 An Advection-Reaction Model

The following system of two semi-linear hyperbolic equations is discussed in both [6] and [12] as "waves travelling in opposite directions":

$$\begin{aligned}\frac{\partial u}{\partial t} &= -\frac{\partial u}{\partial x} - 100uv, \\ \frac{\partial v}{\partial t} &= +\frac{\partial v}{\partial x} - 100uv.\end{aligned}\tag{42}$$

These advection-reaction equations can be used to describe two chemicals with concentrations u and v , which travel through a medium and react when they come into contact.

We set initial and boundary conditions that differ slightly from those in the referenced articles (and we take $x \in [0, 1]$):

$$\begin{aligned}u(x, 0) &= \sin^{100}(\pi(x - 3/4)), \\ v(x, 0) &= \sin^{100}(\pi(x - 1/4)), \\ u(0, t) &= u(1, t) = v(0, t) = v(1, t) = 0.\end{aligned}\tag{43}$$

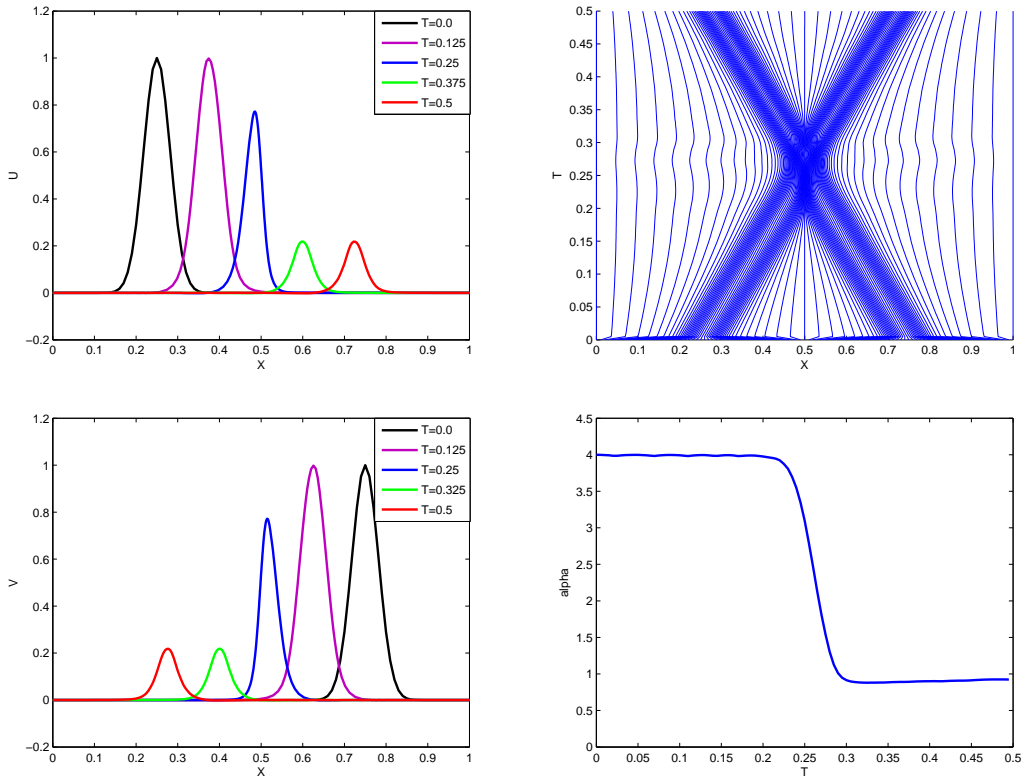


Figure 9: Results for the advection-reaction problem: solution $u(x, t)$ and $v(x, t)$ (top left and bottom left, respectively), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom right). $N = 101$.

Figure 9 shows that the solution of this PDE consists of two wave pulses that start on opposite sides of the centre of the domain, travel towards each

other, decrease in amplitude when they meet and then continue on their way. We note that at time $t = 0.25$, when the waves are interacting (the chemicals are reacting), the pulses are non-symmetric, but at $t = 0.5$ they seem to have regained their symmetry.

The grid movement in figure 9 nicely shows how the grid points follow both pulses. Note that the decrease in amplitude of both pulses causes a decrease in $\alpha(t)$ between $t = 0.2$ and $t = 0.3$. This explains why we find very similar grid point densities outside of the pulses before and after the pulses interact. For the articles we've taken this problem from ([6] and [12]), a constant α was used and both see an increase in the grid point density outside of the pulses after the interaction. It is not immediately clear whether this has any beneficial effects for the solution.

3.6 A Flame Propagation Model

The following PDE system, encountered in [4] and [6], models the mass density u and temperature v of a propagating flame:

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} - uf(v), \\ \frac{\partial v}{\partial t} &= \frac{\partial^2 v}{\partial x^2} + uf(v),\end{aligned}\tag{44}$$

where $f(v) = 3.52 * 10^6 e^{-4/v}$.

We use the following initial and boundary conditions ($x \in [0, 1]$):

$$\begin{aligned}u(x, 0) &= 1, & v(x, 0) &= 0.2, \\ \frac{\partial u}{\partial x}(0, t) &= \frac{\partial v}{\partial x}(0, t) = \frac{\partial u}{\partial x}(1, t) = 0, \\ v(1, t) &= 0.2 + \frac{t}{0.0002}, & 0 \leq t \leq 2 * 10^{-4}, \\ v(1, t) &= 1.2, & 2 * 10^{-4} \leq t \leq 6 * 10^{-3}.\end{aligned}\tag{45}$$

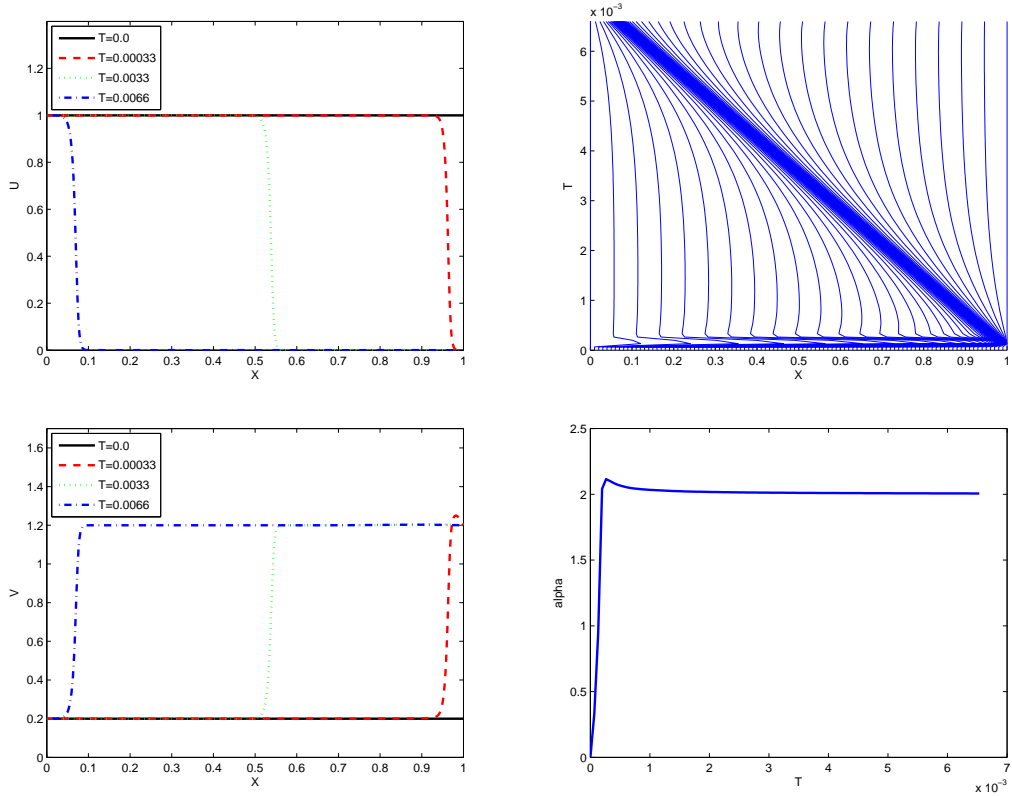


Figure 10: Results for the flame propagation problem: mass density $u(x, t)$ (top left), grid movement $x_i(t)$ (top right), temperature $v(x, t)$ (bottom left) and $\alpha(t)$ (bottom right). $N = 101$.

In figure 10, we see that, initially, mass density u decreases rapidly and temperature v increases rapidly near $x = 1$. At $t = 0.00033$, a front has formed on both u and v , which then travels towards $x = 0$ at an almost constant speed.

This can also be seen in the grid movement plot, which shows the critical area moving in a straight line from the bottom right to the top left.

The adaptivity parameter $\alpha(t)$ starts at 0 and increases to approximately 2.1, while the flame front is forming. Note that the flame front has a constant height while it travels to the left, but at $t = 0.00033$, when it has only just formed, the temperature near $x = 1$ is slightly higher than the constant value it will assume while traveling. This small "bump" is reflected in $\alpha(t)$, which is also approximately constant when the flame propagates from right to left, but has a slightly higher value at $t = 0.00033$.

Because of the very small timescale, we've used a value of 10^{-6} for the temporal smoothing parameter τ in this experiment, instead of the default setting of $\tau = 10^{-3}$.

3.7 A Problem from Combustion

This "combustion" problem is a model of a single reaction ($A \rightarrow B$) with diffusion. Two PDE's are involved for temperature u and mass-fraction v of reactant A:

$$\begin{aligned} Le \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + a D v e^{-\delta/u}, \\ \frac{\partial v}{\partial t} &= \frac{\partial^2 v}{\partial x^2} - D v e^{-\delta/u}. \end{aligned} \quad (46)$$

Heat release $a = 1$, activation energy $\delta = 30.0$, Damkohler number $D = \frac{Re^\delta}{\alpha \delta}$ ($= \frac{\text{reaction rate}}{\text{diffusion rate}}$), Lewis number $Le = 0.9$ and reaction rate $R = 5$ are constants.

We use the following initial and boundary conditions ($x \in [0, 1]$, $t \in [0, T_e]$):

$$\begin{aligned} u(x, 0) &= v(x, 0) = 1, \\ u(1, t) &= v(1, t) = 1, \\ \frac{\partial u}{\partial x}(0, t) &= \frac{\partial v}{\partial x}(0, t) = 0. \end{aligned} \quad (47)$$

A similar problem, though with only one component (the temperature u), is described in [6] and [12].

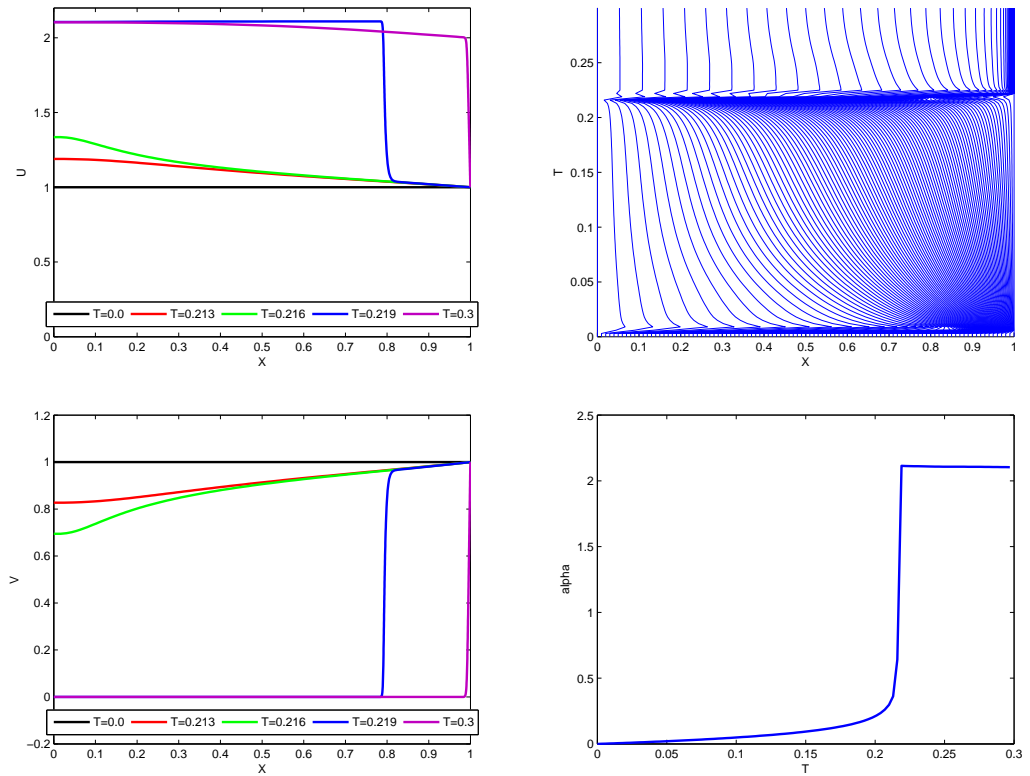


Figure 11: Results for the combustion problem: solution $u(x, t)$ (top left) and $v(x, t)$ (bottom left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom right). $N = 101$.

Figure 11 shows how the temperature at $x = 0$ slowly increases until approximately $t = 0.213$, at which point the substance "ignites": the temperature

rapidly rises to 2.1 and a flame front forms that moves from $x = 0$ to $x = 1$ between $t = 0.213$ and $t = 0.222$. The substance burns itself up: $v = 0$ everywhere the flame front has passed, but $v(x = 1) = 1$ is maintained by the boundary condition. After $t = 0.222$, the solution reaches a "nearly" steady state. Large gradients remain near $x = 1$.

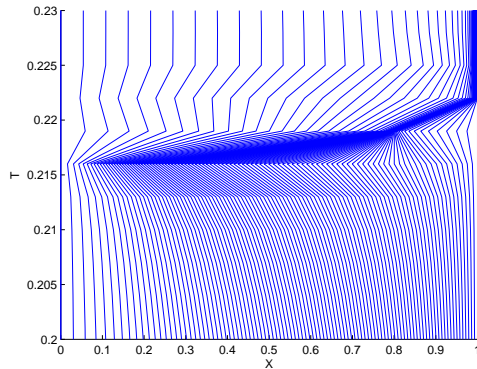


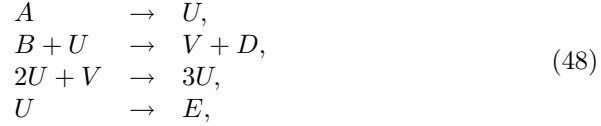
Figure 12: Results for the combustion problem: the grid movement zoomed-in between 0.20 and 0.23. $N = 101$.

We observe that the gridpoints are concentrated somewhat on the right-hand side of the plot before ignition is reached. Close inspection of $u(x, t)$ and $v(x, t)$ for $t < 0.213$ shows that both indeed have larger gradients on the right. These gradients are quite small, but because we use the monitor function (21), the time-dependent $\alpha(t)$ is of the same order of magnitude. The $\alpha(t)$ -plot in figure 11 shows that $\alpha(t)$ starts at 0, increases only slowly at first, then very rapidly around "ignition time" and finally reaches an almost constant value of approximately 2.1. A fixed- α method with a relatively large adaptivity parameter (e.g. $\alpha = 1$) would show a nearly uniform grid for $t < 0.213$.

At ignition, the gridpoints first move very rapidly to the left and then follow the flame front back to the right, as illustrated by the zoomed-in figure 12. In the steady state phase, most gridpoints remain at the right. Note that the grid movement is actually much smoother than figure 12 suggests. Figure 12, like all other figures in this report, is based on data that is written to a file at fixed time-intervals of length $\frac{T_e}{100}$. The actual time-integration dynamically chooses appropriate time-steps during the calculation.

3.8 The Brusselator

The Brusselator model describes a hypothetical tri-molecular reaction, introduced in Brussels in 1971. Solutions found with adaptive grid methods in both 1D and 2D can be found in [11]. The complete model consists of a set of four reactions involving six substances:



which can (with some simplifications) be reduced to a set of two PDEs for the concentrations of U and V :

$$\begin{aligned}
 \frac{\partial u}{\partial t} &= \varepsilon \frac{\partial^2 u}{\partial x^2} + A + u^2 v - (B + 1)u, \\
 \frac{\partial v}{\partial t} &= \varepsilon \frac{\partial^2 v}{\partial x^2} + Bu - u^2 v.
 \end{aligned}
 \tag{49}$$

Constants $A = 1$ and $B = 3.4$ are chemical parameters and $\varepsilon = 10^{-3}$ is the diffusion coefficient.

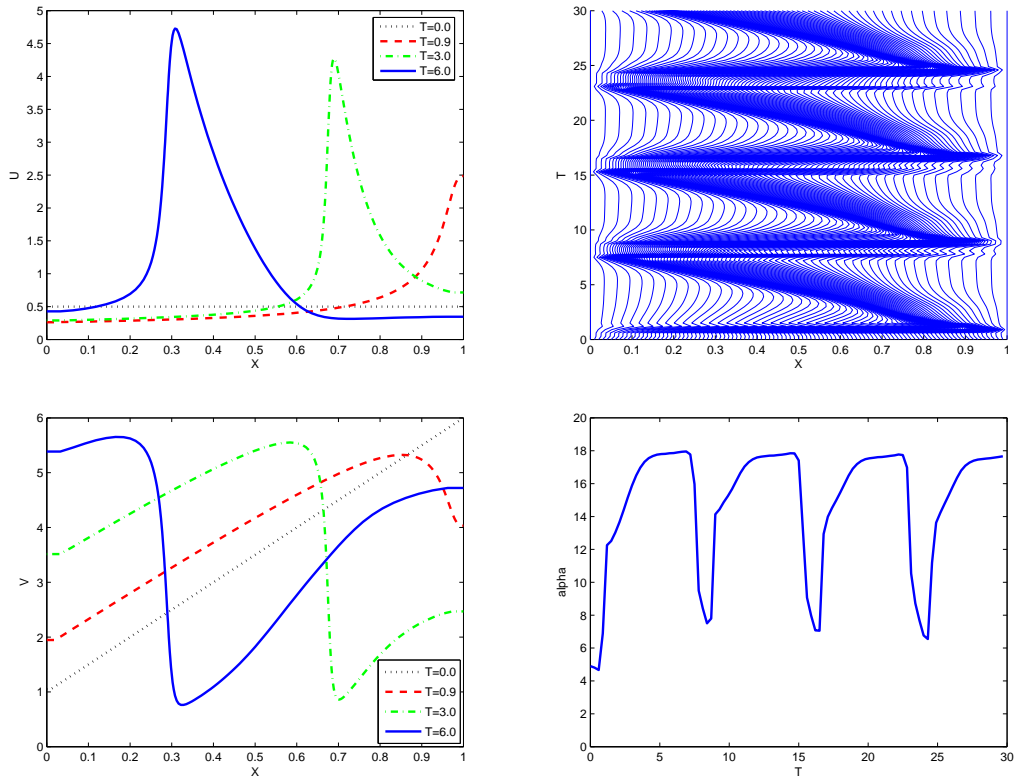


Figure 13: Results for the Brusselator problem: solution $u(x, t)$ and $v(x, t)$ (top left and bottom left, respectively), grid movement $X_i(t)$ (top right) and $\alpha(t)$ (bottom right). $N = 101$.

We use the following initial and boundary conditions ($x \in [0, 1]$ and $t \in [0, T_e]$):

$$\begin{aligned} u(x, 0) &= 0.5, \\ v(x, 0) &= 1 + 5x, \\ \frac{\partial u}{\partial x}(0, t) &= \frac{\partial u}{\partial x}(1, t) = 0, \\ \frac{\partial v}{\partial x}(0, t) &= \frac{\partial v}{\partial x}(1, t) = 0. \end{aligned} \tag{50}$$

We've used the adaptive grid method with time-dependent $\alpha(t)$ to solve the Brusselator problem with these ICs and BCs and $T_e = 30.0$. The results are plotted in figure 13.

Both u and v experience a wave that periodically moves from right to left. This periodicity is not immediately apparent from the $u(x, t)$ -plot in figure 13, but it is very nicely visible in both the grid movement and the evolution of $\alpha(t)$. The overall behaviour of $u(x, t)$ and $v(x, t)$ is perhaps best illustrated by figure 14.

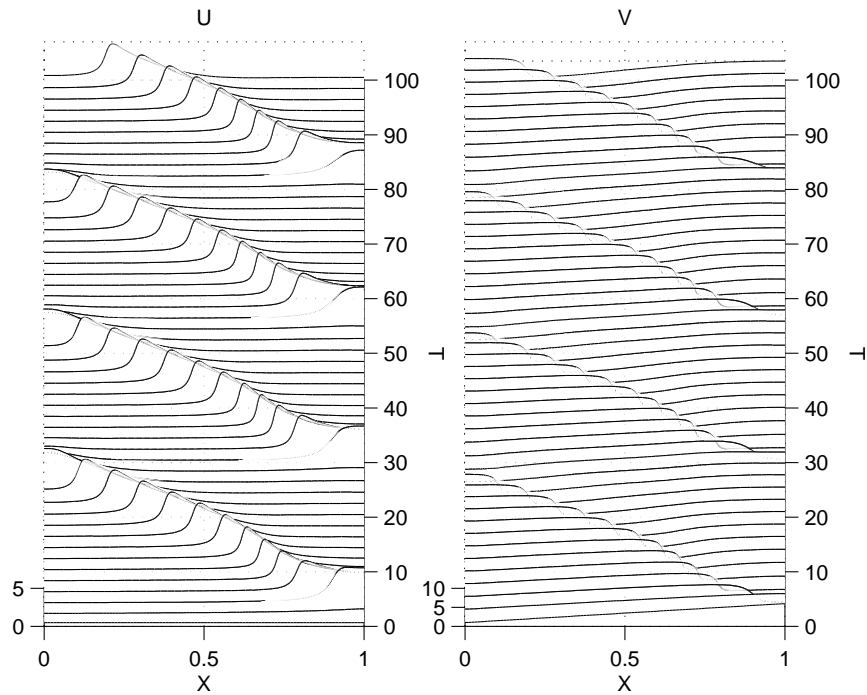


Figure 14: Results for the Brusselator problem: solution $u(x, t)$ (left) and $v(x, t)$ (right). $N = 101$.

3.9 The FitzHugh-Nagumo Equations

The FitzHugh-Nagumo equation is a simplification of the Hodgkin-Huxley model, which models the control of the electrical potential across cell membrane [21]. In the spatially distributed FitzHugh-Nagumo model, diffusion is added to the general FitzHugh-Nagumo model. We've tested the following specific case, also seen in [4]:

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + f(u) - v, \\ \frac{\partial v}{\partial t} &= b(u - cv),\end{aligned}\tag{51}$$

where $f(u) = u(u - a)(1 - u)$ and $a = 0.139$, $b = 0.008$ and $c = 2.54$. The initial and boundary conditions ($x \in [0, 120]$) we've used are:

$$\begin{aligned}\frac{\partial u}{\partial x}(0, t) &= -\frac{I}{2}, \\ \frac{\partial u}{\partial x}(120, t) &= 0, \\ u(x, 0) &= v(x, 0) = 0,\end{aligned}\tag{52}$$

with $I = 0.45$.

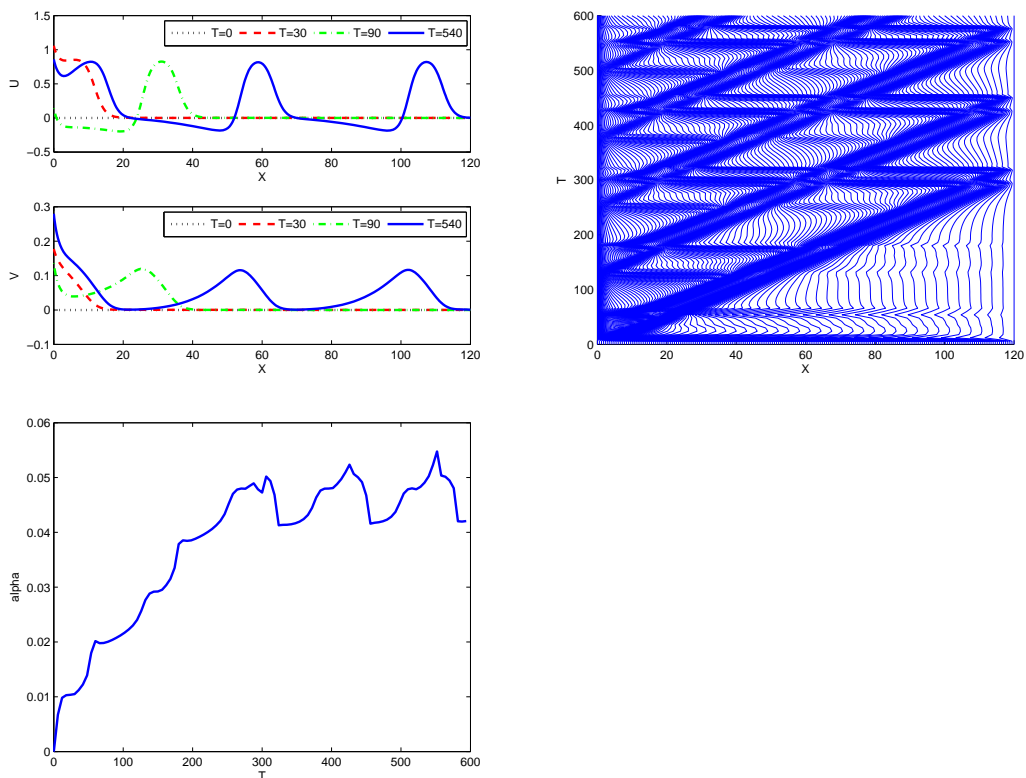


Figure 15: Results for the FitzHugh-Nagumo problem: solution $u(x,t)$ and $v(x,t)$ (top left), grid movement $x_i(t)$ (top right), and $\alpha(t)$ (bottom left). $N = 201$.

Figure 15 shows that the solution consists of pulses on both $u(x,t)$ and $v(x,t)$ that appear at the left and move to the right, where they disappear

again. Eventually, up to three different pulses on $u(x, t)$ and $v(x, t)$ can be present at a time. The moving grid follows each pulse. While $\alpha(t)$ is zero at $t = 0$, it increases as pulses start to appear. After the maximum number of three pulses between $x = 0$ and $x = 120$ is reached, $\alpha(t)$ oscillates as new pulses appear on the left and old ones disappear on the right.

Note that the domain for this problem ($x \in [0, 120]$) was much larger than for most other problems we've tested ($x \in [0, 1]$, usually). This makes the $(x_R - x_L)$ -term in $\alpha(t) = \frac{\int_{x_L}^{x_R} |\frac{\partial u}{\partial x}| dx}{(x_R - x_L)}$ much more important. At first, we performed tests with $\alpha(t) = \int_{x_L}^{x_R} |\frac{\partial u}{\partial x}| dx$, which gave a value of α that was much too large, resulting in a nearly uniform grid for this FitzHugh-Nagumo problem.

3.10 The Gray-Scott Reaction-Diffusion System

In this example, we study the Gray Scott equations, which model the following chemical reactions (see [10]):



in a gel reactor coupled to a reservoir in which concentrations of U and V are maintained constant. This system is described by the following pair of reaction-diffusion equations:

$$\begin{aligned} \frac{\partial u}{\partial t} &= D_U \frac{\partial^2 u}{\partial x^2} - uv^2 + A(1 - u), \\ \frac{\partial v}{\partial t} &= \varepsilon D_U \frac{\partial^2 v}{\partial x^2} + uv^2 - Bv, \end{aligned} \quad (54)$$

where u and v are the concentrations of U and V , respectively, $A = 2.0\varepsilon$ denotes the rate at which U is fed from the reservoir into the reactor, $B = A + k = 0.4\varepsilon^{\frac{1}{3}}$ is the sum of A and the rate k at which V is converted to inert product P , $D_U = \frac{10^{-4}}{16}$ is the diffusivity of U and $\varepsilon = \frac{D_V}{D_U} = 0.01$ is the ratio between the diffusivities of V and U .

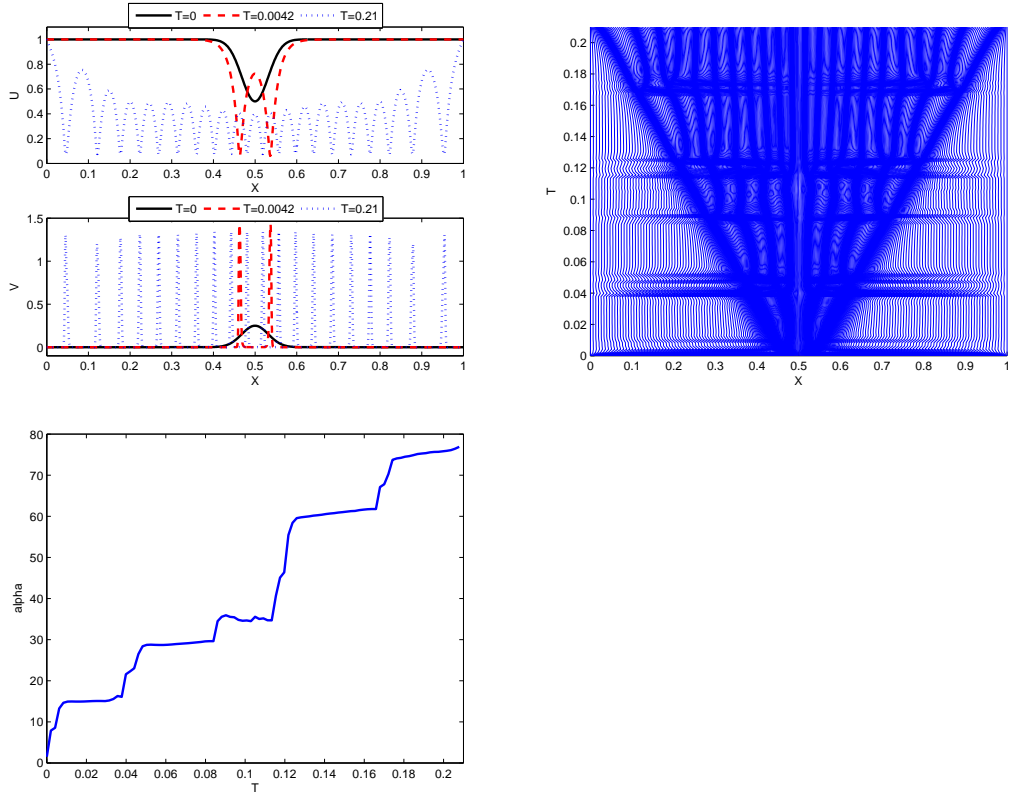


Figure 16: Results for the Gray-Scott reaction-diffusion system: solution $u(x, t)$ and $v(x, t)$ (top left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 801$.

For the above equations (54), the concentrations of U and V in the reservoir are assumed to be one and zero, respectively. This is why U is fed from the reservoir into the reactor, while V is siphoned out. It is also reflected in the boundary conditions. We've tested with the following initial and boundary conditions:

$$\begin{aligned} u(x, 0) &= 1 - \frac{1}{2} \sin^{100}(\pi x), & u(0, t) &= u(1, t) = 1, \\ v(x, 0) &= 1 - \frac{1}{4} \sin^{100}(\pi x), & v(0, t) &= v(1, t) = 0. \end{aligned} \quad (55)$$

We've used the same rescaled spatial variable \tilde{x} as in [10], such that $\tilde{x} \in [0, 1]$. Furthermore, we've used a rescaled temporal variable $\tilde{t} = 10^{-5}t$, such that our choice of $T_e = 0.21$ means that the outermost pulses of the solution almost reach the boundaries (see figure 16) at T_e . The corresponding T_e in [10] would be 21000.

Figure 16 shows the results of a run with 801 moving gridpoints. We observe a pattern similar to the results found in [10]: a single initial pulse splits into two pulses, which move in opposite directions towards the boundaries and, as more and more splits occur, the area between the two is filled up with pulses. The outermost pulses move at an apparently constant speed, which is the same in our results as in [10], if we take the rescaling of the temporal variable into account.

However, there are also differences. Whereas Doelman et al. report a total number of 20 pulses at $t = 2 \times 10^4$, we find 22 pulses at the equivalent time $\tilde{t} = 0.2$. We know, both from [10] and our own experiments, that lower numerical accuracy (due to a lower number of gridpoints) can lead to less pulse splittings and a lower final number of pulses. It is tempting to conclude that our higher number of pulses at $\tilde{t} = 0.2$ is evidence of a better numerical accuracy. However, we should point out the fact that at the earlier time $\tilde{t} = 0.1$, our results show only 10 pulses, while Doelman et al. have 12 at $t = 1 \times 10^4$. Possibly, this 'non-occurrence' of one splitting just before $\tilde{t} = 0.1$ in our experiments creates the required 'breathing space' for the two additional splittings just before $\tilde{t} = 0.2$.

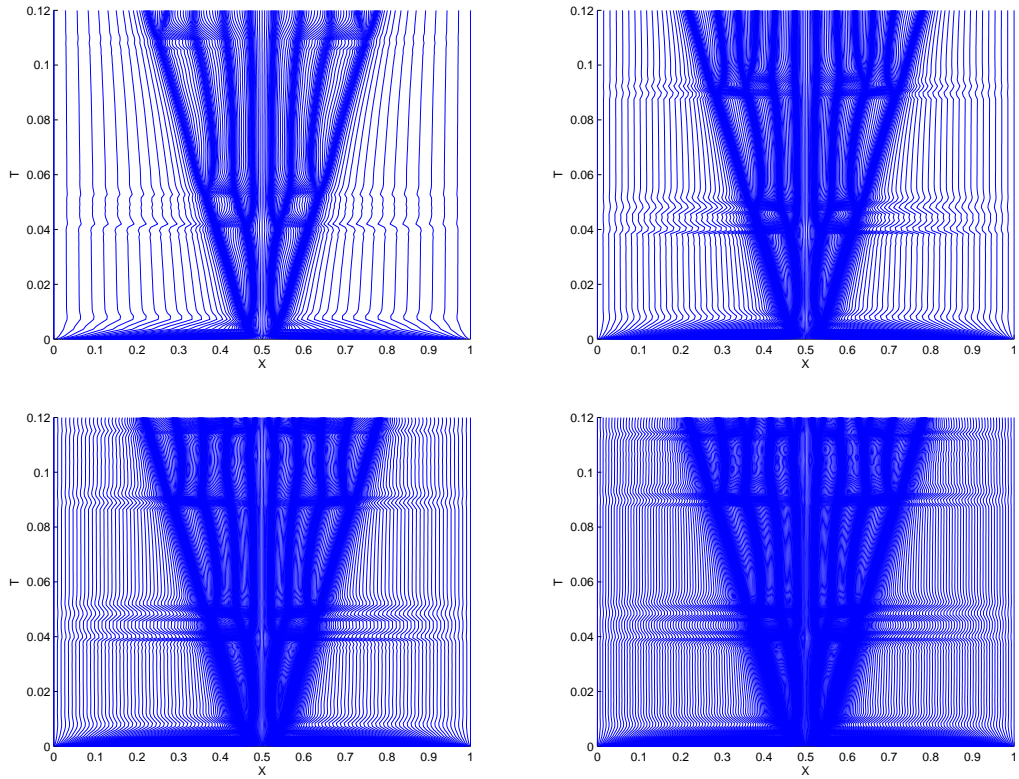


Figure 17: Moving grid results for the Gray-Scott reaction-diffusion system with varying numbers of gridpoints: $N = 201$ (top left), $N = 401$ (top right), $N = 601$ (bottom left), $N = 801$ (bottom right).

Figure 17 shows how the number of pulses at $T = 0.1$ converges towards 10 as we increase the number of gridpoints N . Note that we observe 8 pulses for $N = 201$ and 12 for $N = 401$, but 10 for $N = 601$ and $N = 801$ proving that higher numerical accuracy (as a result of a larger number of gridpoints) does not necessarily lead to a higher (or even at least equal) number of pulses in the solution.

3.11 A Cylindrical Reaction-Diffusion PDE

This example is the scalar reaction-diffusion equation in cylindrical coordinates ($m = 1$), that we've taken from [5] (Example II). This equation describes the temperature $T(r, z)$ in a cylinder, where r is the radial direction and z the axial direction. The latter can be treated as a timelike coordinate. We perform a straightforward substitution ($u = T$, $x = r$ and $t = z$ with $x \in [0, 1]$ and $t \in [0, 1]$) to work with our usual variable names and the PDE becomes:

$$\frac{\partial u}{\partial t} = \frac{1}{x} \frac{\partial}{\partial x} \left(\beta x \frac{\partial u}{\partial x} \right) + \gamma e^{\frac{u}{1+\varepsilon u}}, \quad (56)$$

where $\beta = 0.1$, $\gamma = 1.0$ and $\varepsilon = 0.1$ are constants that represent given thermal properties.

The initial and boundary conditions are:

$$u(x, 0) = 0, \quad \frac{\partial u}{\partial x}(0, t) = 0, \quad u(1, t) = 0. \quad (57)$$

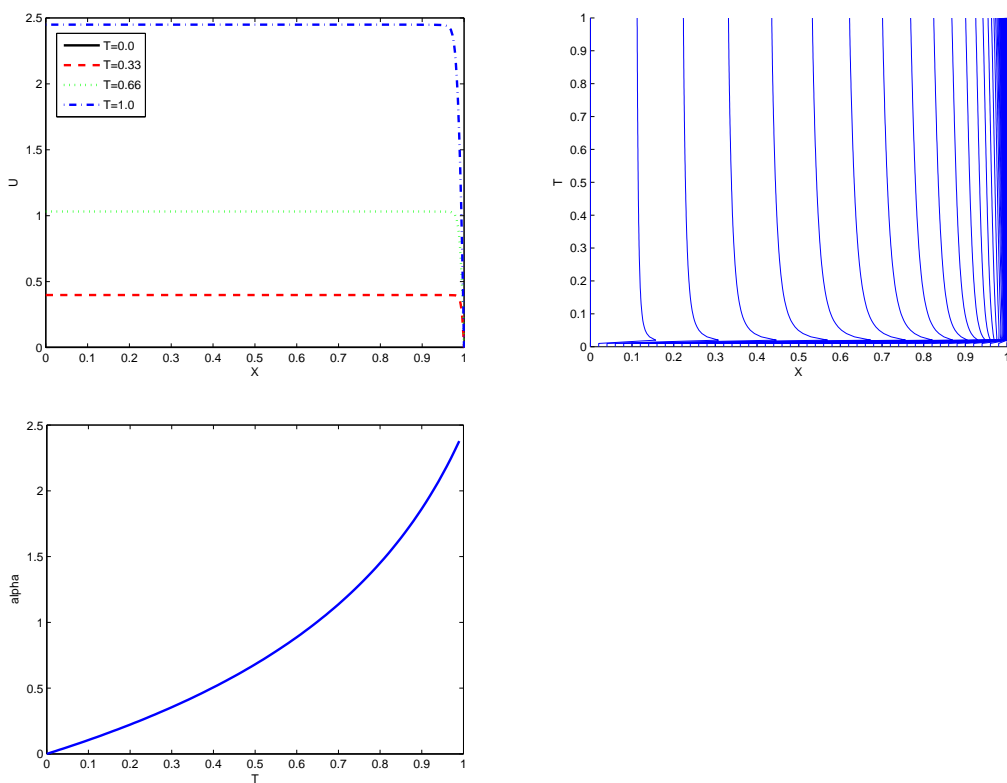


Figure 18: Results for the cylindrical reaction-diffusion problem: solution $u(x, t)$ (top left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 51$.

Figure 18 shows that the temperature of the cylinder u , for a given value of the axial coordinate t , is constant along the radial direction x , except at

the very edge (near $x = 1.0$), where it rapidly drops to zero. Along the axial direction t , the temperature u increases more gradually, from 0.0 at $t = 0.0$ to 2.5 at $t = 1.0$. Note that, since $u(x, t)$ is monotonically decreasing with respect to x and always 0.0 at $x = 1.0$, $\alpha(t)$ simply shows the temperature on the axis: $u(0, t)$.

The grid movement is especially interesting when compared to the results in [5], where the gridpoints always keep moving towards $x = 1.0$, where the temperature front is. Due to the fact that our time-dependent adaptivity parameter $\alpha(t)$ increases with t , we observe gridpoints that, after starting in a uniform distribution, move rapidly towards the critical area at low values of t (i.e. when $\alpha(t)$ is still small), and then slowly move back to the left to form a slightly more evenly distributed grid as t (and $\alpha(t)$) increase.

3.12 Heat Flow of Harmonic Maps from Surfaces

In this chapter, we deal with a special case of the harmonic map heat flow equation. Theoretical aspects of this problem are discussed in [19], while earlier numerical results obtained with a moving grid method are reported in [13]. The general equation is as follows:

$$\frac{\partial H}{\partial t} = \Delta H + |\nabla H|^2 H, \quad (58)$$

where $H(\mathbf{x}, t)$ denotes a unit vector in \mathbb{R}^3 , \mathbf{x} being a vector in $\Omega \subset \mathbb{R}^N$ and $|\nabla H|^2 = \sum_{j=1}^N \sum_{i=1}^3 \left(\frac{\partial H_i}{\partial x_j}\right)^2$. Stationary solutions of (58) are harmonic maps from Ω to the 2-sphere S^2 .

For the special case where Ω is a disk in \mathbb{R}^2 , using polar coordinates (r, ϕ) and assuming radial symmetry, solutions of (58) will have the following form:

$$H(r, \phi, t) = \begin{pmatrix} \cos \phi \sin \theta(r, t) \\ \sin \phi \sin \theta(r, t) \\ \cos \theta(r, t) \end{pmatrix}, \quad (59)$$

where $\theta(r, t)$ satisfies the following PDE:

$$\frac{\partial \theta}{\partial t} = \frac{\partial^2 \theta}{\partial r^2} + \frac{1}{r} \frac{\partial \theta}{\partial r} - \frac{\sin(2\theta)}{2r^2}. \quad (60)$$

This is the equation we will attempt to solve numerically. For consistency of notation, we will once again label the solution function u and the spatial variable x , so we substitute $u = \theta$ and $x = r$ into (60):

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{1}{x} \frac{\partial u}{\partial x} - \frac{\sin(2u)}{2x^2}. \quad (61)$$

We use the following initial and boundary conditions (for $x \in [0, 1]$ and $t \in [0, T_e]$):

$$u(x, 0) = 8.5 \frac{\pi}{2} x^2, \quad u(0, t) = 0, \quad u(1, t) = 8.5 \frac{\pi}{2}. \quad (62)$$

Since we numerically solve this PDE in its "m = 1"-form (i.e. with cylindrical/polar coordinates), the boundary condition at $x = 0$ is, in practice, replaced by the symmetry condition.

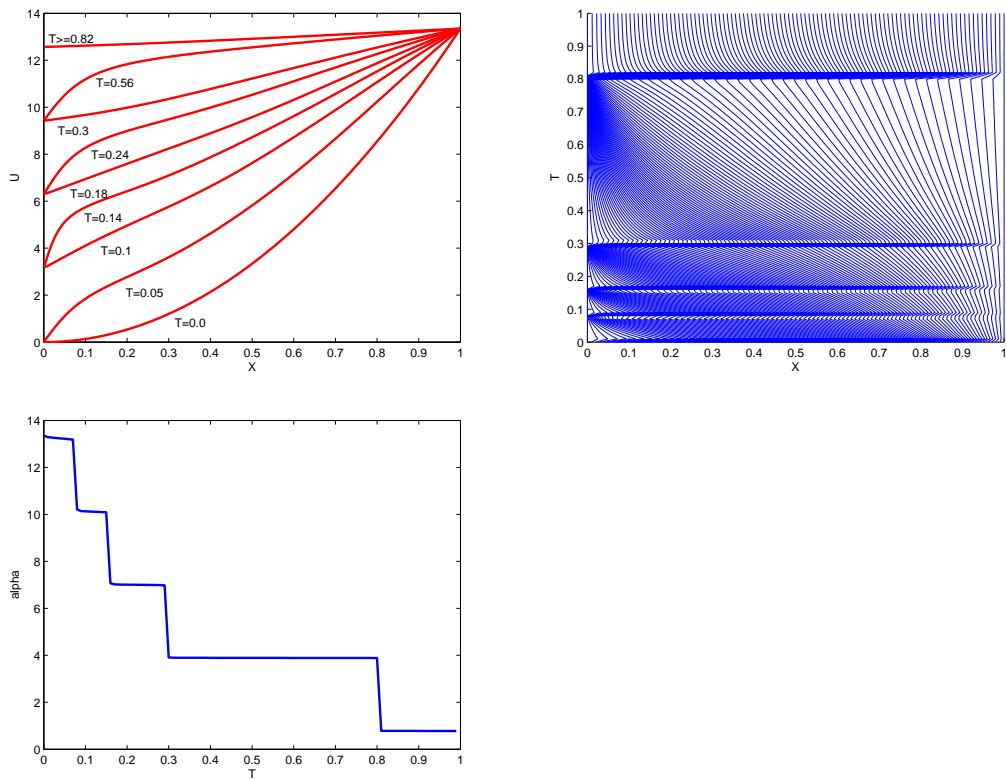


Figure 19: Results for the "heat flow of harmonic maps from surfaces" model: solution $u(x, t)$ (top left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 101$.

In figure 19, we observe a repeating pattern in the behaviour of $u(x, t)$. For some time, the value of $u(0, t)$ remains fixed, while u between $x = 0$ and $x = 1$ increases. At some point in time, the value of $u(0, t)$ suddenly increases. The size of each jump is very close to π , which is in agreement with the analysis in [19]. The times at which these jumps in $u(0, t)$ take place can easily be seen in the $\alpha(t)$ -plot. Figure 19 also shows how the gridpoints move towards the left border during each period of constant $u(0, t)$ and then quickly go back to an almost uniform distribution after each jump in $u(0, t)$.

3.13 A Basic Test Problem in Spherical Coordinates

This example is simply meant to test our code on a problem with spherical coordinates ($m = 2$). We've taken example problem 2.4 from [8], which has the very basic $m = 2$ equation:

$$\frac{\partial u}{\partial t} = \frac{1}{x^2} \frac{\partial}{\partial x} \left(x^2 \frac{\partial u}{\partial x} \right), \quad (63)$$

with $x \in [0, 1]$ and $t \in (0, 0.8]$.

The initial condition is $u(x, 0) = x^2$ and the boundary conditions are the symmetry condition at $x = 0$ and:

$$u(1, t) = 1 + 6t. \quad (64)$$

This problem has a known exact solution $u_{ex}(x, t)$:

$$u_{ex}(x, t) = x^2 + 6t. \quad (65)$$

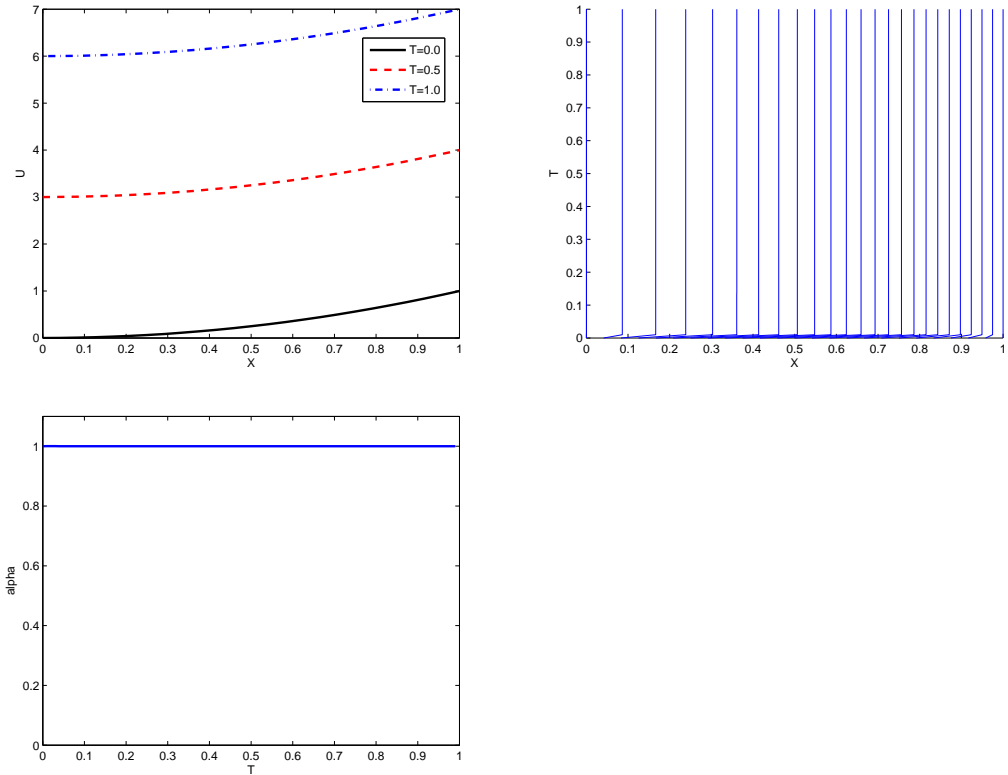


Figure 20: Results for the simple $m = 2$ problem: solution $u(x, t)$ (top left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 25$.

Figure 20 shows that our solver reproduces the exact solution $u_{ex}(x, t)$ quite well. The grid quickly takes on a constant, non-uniform distribution and α is constant throughout the run. We conclude that, while our method was able to

solve this problem, it is not a problem that requires an adaptive grid method to solve (although a non-uniform fixed grid might still give better results than a uniform one).

3.14 The Implosion Model in Spherical Coordinates

With this example, we want to try and solve a slightly more difficult problem with spherical coordinates ($m = 2$). We generate such a problem by converting the implosion problem (Section 3.1) to its equivalent in spherical coordinates. We keep the same exact solution $u_{ex}(x, t) = e^{-(e^t x)^2}$ and choose a function $f(x, t)$, such that the following general $m = 2$ -PDE has $u_{ex}(x, t)$ as its solution:

$$\frac{\partial u}{\partial t} = \frac{1}{x^2} \frac{\partial}{\partial x} \left(x^2 \frac{\partial u}{\partial x} \right) + f(x, t). \quad (66)$$

Consequently, we need:

$$f(x, t) = \frac{\partial u_{ex}}{\partial t} - \frac{1}{x^2} \frac{\partial}{\partial x} \left(x^2 \frac{\partial u_{ex}}{\partial x} \right). \quad (67)$$

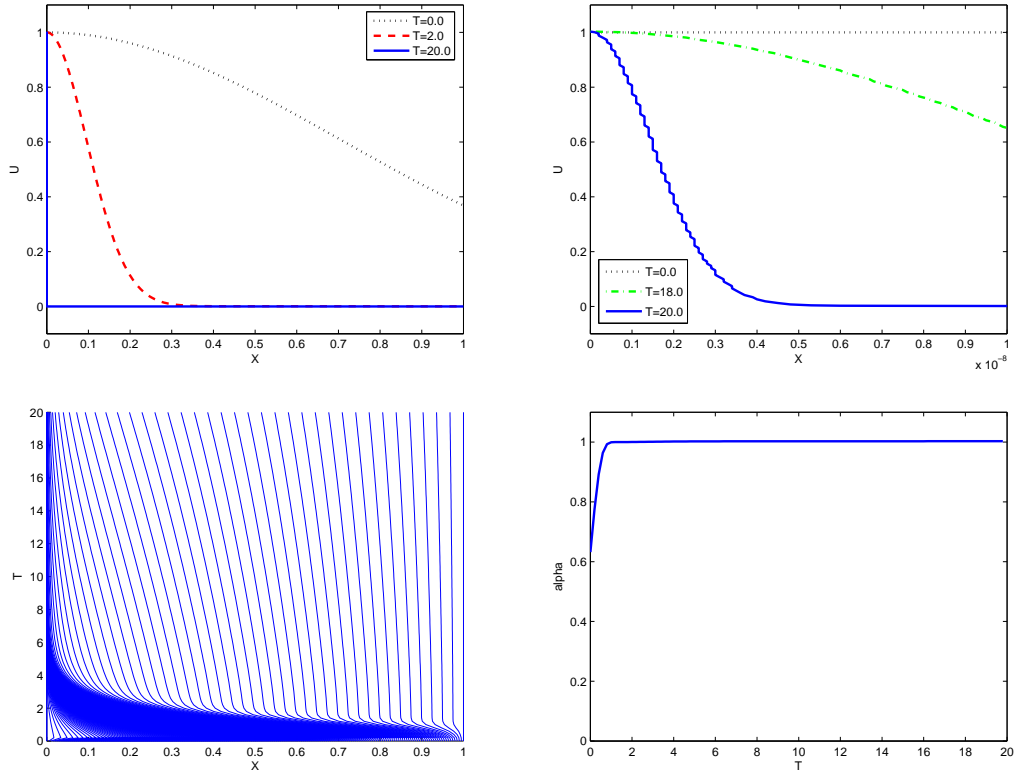


Figure 21: Results for the implosion problem with spherical coordinates: solution $u(x, t)$ (top left), a zoomed-in version of $u(x, t)$ (top right), grid movement $x_i(t)$ (bottom left) and $\alpha(t)$ (bottom right). $N = 201$.

The initial and boundary conditions are derived from $u_{ex}(x, t)$, except for the boundary condition at $x = 0$, which is the symmetry condition inherent to working with spherical coordinates.

$$u(x, 0) = e^{-x^2}, \quad u(1, t) = e^{-(e^t)^2}, \quad \frac{\partial u}{\partial x}(0, t) = 0. \quad (68)$$

Note that we use a slightly different domain ($x \in [0, 1]$) than we did in section 3.1.

As figure 21 shows, the adaptive grid method with time-dependent adaptivity parameter is quite capable of solving the implosion problem with spherical coordinates. Even at time $T_e = 20.0$, the solution still looks like a (very narrow) Gaussian function with appropriate height. We do notice that, when zoomed in closely, the solution appears somewhat jagged at $T_e = 20.0$. It is possible that we've encountered a numerical precision limit that causes a minimum distance between consecutive gridpoints.

The grid points nicely move towards the critical area, much like they did in section 3.1. Note that $\alpha(t)$ has a constant value of 1 throughout most of the run, but starts out slightly lower. This is due to the initially non-zero boundary condition at $x = 1$.

3.15 A Brine Transport Model

This model, which is discussed in detail in [9], describes the transport of pollutants by groundwater flow, when released from a repository in a rock salt formation. It can be formulated as a system of two PDEs with pressure p and salt concentration ω as independent variables. The equations to be solved are (with $x \in [0, L]$ and $t \in [0, T_e]$):

$$\begin{aligned} n\rho\beta\frac{\partial p}{\partial t} + n\rho\gamma\frac{\partial \omega}{\partial t} &= -\frac{\partial}{\partial x}(\rho q), \\ n\rho\frac{\partial \omega}{\partial t} &= -\rho q\frac{\partial \omega}{\partial x} - \frac{\partial}{\partial x}(\rho J), \end{aligned} \quad (69)$$

with the following initial conditions:

$$\omega(x, 0) = 0, \quad p(x, 0) = p_0[(1 - x/L)p_{left} + (x/L)p_{right}], \quad (70)$$

and boundary conditions:

$$\begin{aligned} \omega(0, t) &= \omega_0 > 0, & \frac{\partial \omega}{\partial x}(L, t) &= 0, \\ p(0, t) &= p_0 p_{left}, & p(L, t) &= p_0 p_{right}, \end{aligned} \quad (71)$$

where the fluid velocity $q = -\frac{k}{\mu}(\frac{\partial p}{\partial x} + \rho g)$, $J = -\lambda|q|\frac{\partial \omega}{\partial x}$ and $\rho = \rho_0 e^{\beta(p-p_0) + \gamma\omega}$. The reference density ρ_0 , reference pressure p_0 , compressibility coefficient β and salt coefficient γ are all constants. We use a re-scaling of the problem, normalising many of these coefficients, such that: $n = p_0 = \rho_0 = k = \mu = \omega_0 = L = 1.0$, while $\beta = 1.0E - 5$, $\gamma = 0.1794$ and $g = 0.0981$. We test three examples with different values of λ , p_{left} and p_{right} .

We prefer to use the labels u and v for our numerical solutions, so we simply substitute $u = p$ and $v = \omega$. We investigate the same three examples discussed in [9].

3.15.1 Example I

For the first example, we choose $\lambda = 0.001$, $T_e = 5$, $p_{left} = 1.7$ and $p_{right} = 1.0$.

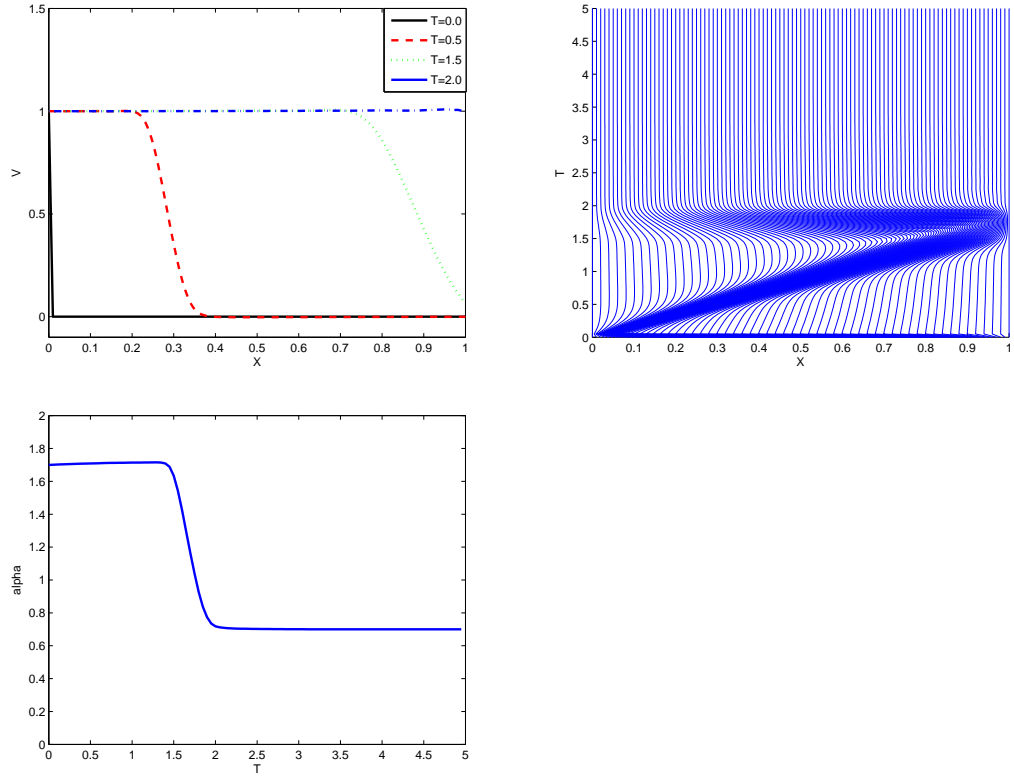


Figure 22: Results for the brine transport model, example I: salt concentration $v(x, t)$ (top left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 101$.

In figure 22, we see a front move from the left border to the right. Once it reaches the right border, a nearly constant function $v = 1$ remains. The grid moves with the front in the early phase and returns to a uniform distribution once the constant function is reached. The adaptivity parameter starts at a value around 1.7 and ends around 0.7. The contribution of the salt concentration v to $\alpha(t)$ at that point is close to zero. We conclude that $\alpha(t)$ at $t > 2.0$ is almost completely determined by the pressure u , which we do not show here.

3.15.2 Example II

For the second example, we choose $\lambda = 0.0001$, $T_e = 500$, $p_{left} = 1.11$ and $p_{right} = 1.0$. The results are shown in figure 23.

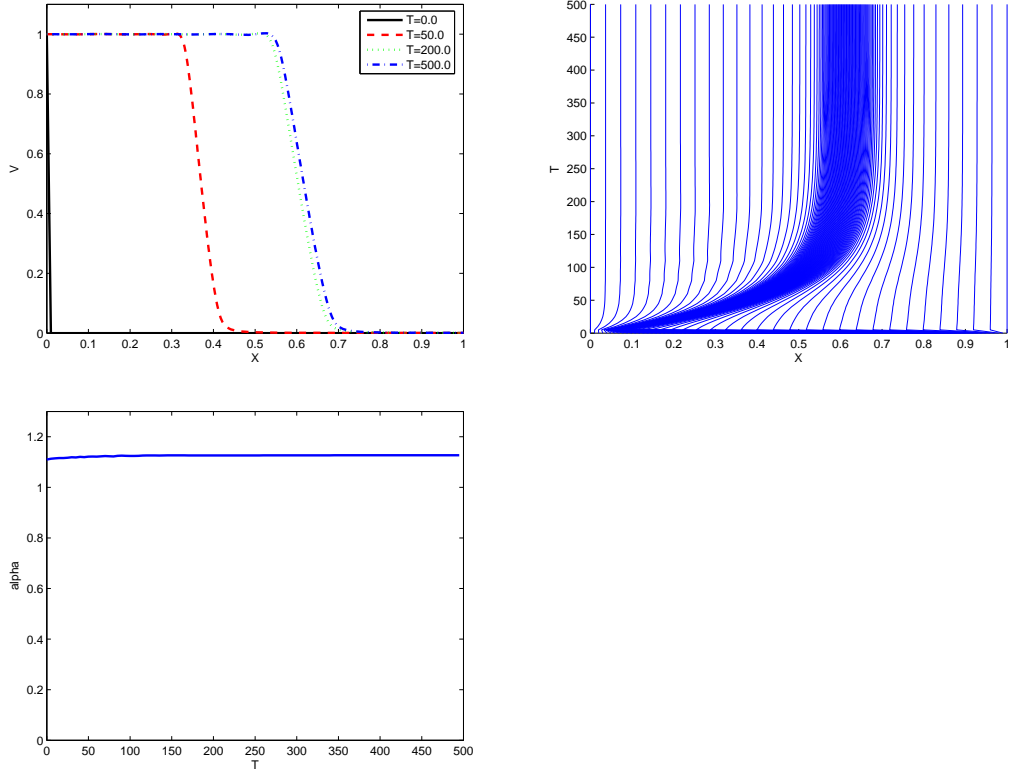


Figure 23: Results for the brine transport model, example II: salt concentration $v(x, t)$ (top left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 101$.

In figure 23, we again see a rightward travelling front. Due to the smaller initial pressure gradient, this front moves slowly compared to example 1, which explains the need for a larger time-scale. Furthermore, this front does not reach the right border, but comes to a standstill near $x = 0.6$, resulting in a steady-state with a non-uniform salt concentration. The fact that the strong gradient in v stays throughout the run also explains why α remains constant.

3.15.3 Example III

For the third example, we use the same values $\lambda = 0.001$, $T_e = 5$, $p_{left} = 1.7$ and $p_{right} = 1.0$ as in example I, but the boundary condition $\omega(0, t) = 1$ is replaced by the step function:

$$\begin{aligned} \omega(0, t) &= 1, & t \in [0, 0.75], \\ \omega(0, t) &= 0, & t \in (0.75, T_e], \end{aligned} \quad (72)$$

which we approximate by $\omega(0, t) = 0.5 * (1.0 - \tanh(1000.0 * (t - 0.75)))$ in practice. The results are shown in figure 24.

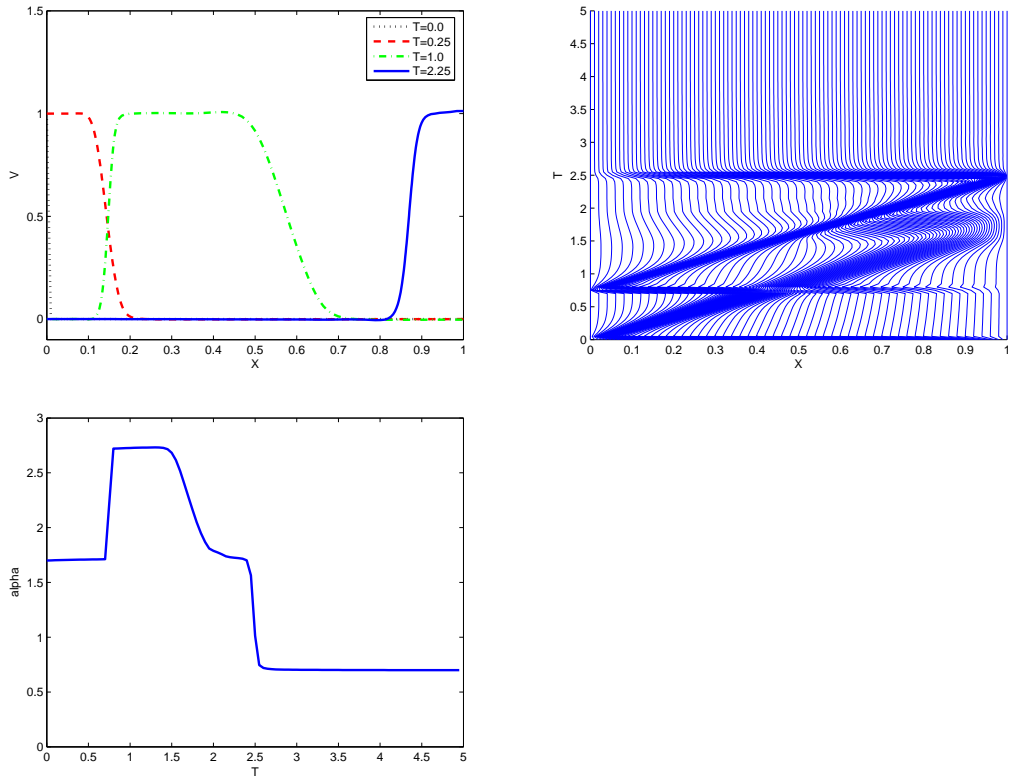


Figure 24: Results for the brine transport model, example III: salt concentration $v(x, t)$ (top left), grid movement $x_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 101$.

As we see in figure 24, this third example features a "pulse" on the salt concentration v that moves from the left border to the right and then disappears. The final result is a uniform salt concentration of $v = 0$. The behaviour of $\alpha(t)$ can be described as follows: Throughout the run, there is a basic "minimum" contribution of approximately 0.7 from the pressure gradient. Initially, the right slope of the pulse on v contributes an additional 1.0 and around $t = 0.7$, the left slope contributes another 1.0. Then, when the pulse reaches the right border around $t = 1.5$, first the right slope disappears and around $t = 2.5$ also the left slope is gone.

3.16 A Model Describing Tumour Angiogenesis

This is a problem from medicine [20], with the following PDEs (for $x \in [0, 1]$ and $t \in [0, T_e]$):

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial}{\partial x} \left(10^{-3} \frac{\partial u}{\partial x} - \left[\frac{3}{4} \frac{\partial v}{\partial x} \right] u \right) - 4u + 10^2 u(1-u) \max(0, v-0.2), \\ \frac{\partial v}{\partial t} &= \delta \frac{\partial^2 v}{\partial x^2} - v - 10 \frac{uv}{1+v},\end{aligned}\tag{73}$$

where u denotes the density of endothelial cells (blood) and v is the so-called Tumour Angiogenesis Factor (TAF). The δ in the equation for v is the diffusion parameter. In [13], numerical solutions are shown for this problem, with the following initial and boundary conditions:

$$\begin{aligned}u(x, 0) &= 0, & \text{if } 0 \leq x < 1, \\ u(x, 0) &= 1, & \text{if } x = 1, \\ v(x, 0) &= \cos\left(\frac{1}{2}\pi x\right), \\ u(0, t) &= v(1, t) = 0, \\ u(1, t) &= v(0, t) = 1.\end{aligned}\tag{74}$$

We've tried to solve this PDE numerically for two different values of the diffusion parameter δ .

3.16.1 Example I

In the first example, we use $\delta = 0.001$.

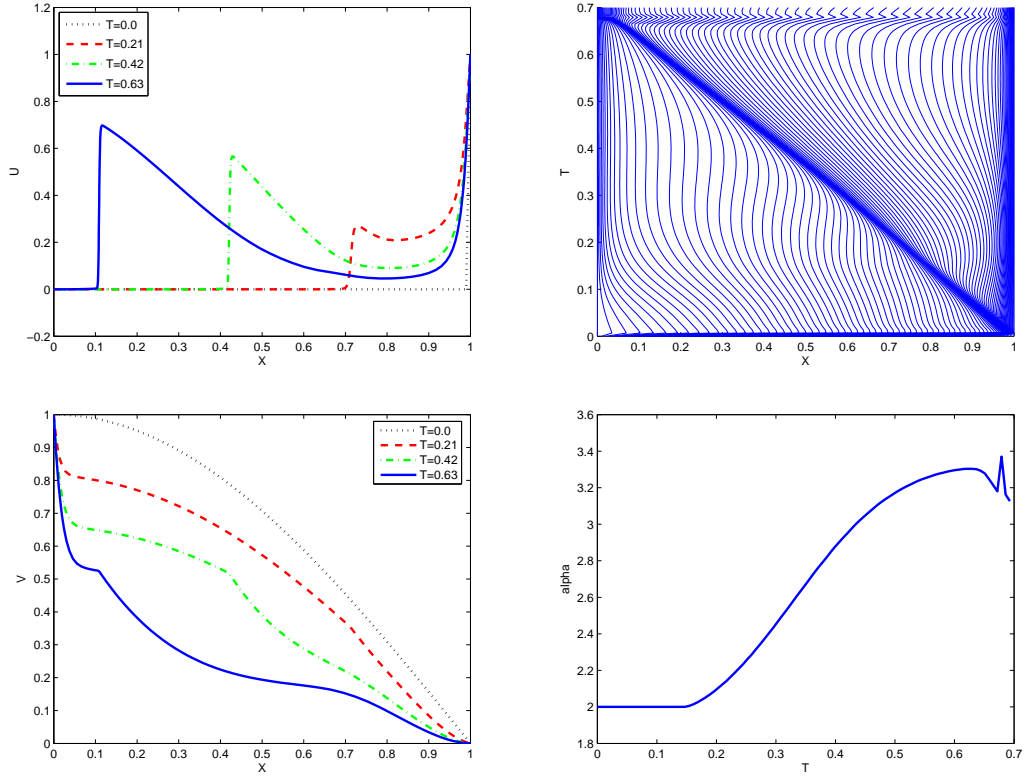


Figure 25: Results for the tumour angiogenesis model, example I: endothelial cell density $u(x, t)$ (top left), grid movement $x_i(t)$ (top right), tumour angiogenesis factor $v(x, t)$ (bottom left) and $\alpha(t)$ (bottom right). $N = 101$.

In figure 25, we see that the endothelial cell density $u(x, t)$, which has a jump at $x = 1.0$ from the start, quickly develops a second, smaller "front", which moves from right to left, gaining height as it travels. In the space between these two large gradients, $u(x, t)$ gradually decreases. The moving grid follows the moving front from right to left, while maintaining a high gridpoint density near the right border as well. Apart from some oscillation near $t = T_e$, $\alpha(t)$ increases steadily throughout the run.

The tumour angiogenesis factor $v(x, t)$ decreases throughout the run. The most notable feature on $v(x, t)$ is perhaps the sharp bend at the same x as the travelling front on $u(x, t)$. Note that $v(x, t)$ is a monotonically decreasing function of x at all times, with fixed values at the boundaries. Consequently, its contribution to $\alpha(t)$ is constant. The plot of $\alpha(t)$ shows a constant value of 2 until approximately $t = 0.16$. We conclude that $u(x, t)$, which starts as a monotonically increasing function of x , becomes nonmonotonic at that point in time. Apparently, we can read the approximate time at which the travelling front on $u(x, t)$ becomes a "bump" from the $\alpha(t)$ -plot.

3.16.2 Example II

In the second example, we use $\delta = 1.0$.

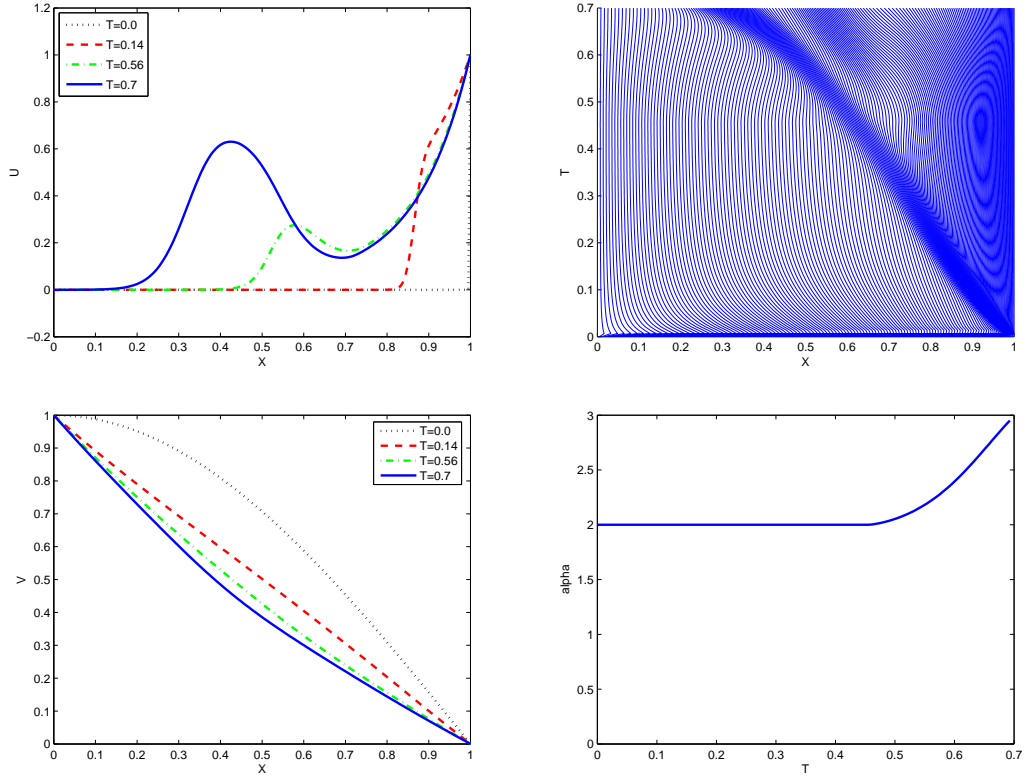


Figure 26: Results for the tumour angiogenesis model, example II: solution $u(x, t)$ (top left), grid movement $x_i(t)$ (top right), solution $v(x, t)$ (bottom left) and $\alpha(t)$ (bottom right). $N = 201$.

The results in figure 26 show a much smoother u than those in figure 25, due to the higher diffusion parameter δ . Once again, we see a bumpy feature travel from right to left, increasing in height as it travels. The travelling speed of the left foot of this bump turns out to be very dependent on the numerical accuracy: if we change N from 201 (used for figure 26) to 101, it reaches the left border before $t = T_e$. Higher values of N (we also tested 401 and 801) make the left foot move even more slowly, but the difference becomes less pronounced. The top of the bump moves at the same speed for all values of N that we've tested.

The tumour angiogenesis factor $v(x, t)$ decreases both more slowly and more smoothly in this example than in the first.

3.17 A Gas Discharge Model

In [16] we encounter the following model for simulating the dynamical development of streamers out of a macroscopic initial ionization seed in a so-called non-attaching gas:

$$\frac{\partial n_e}{\partial t} + \frac{\partial j_e}{\partial x} = S, \quad (75)$$

$$\frac{\partial n_+}{\partial t} + \frac{\partial j_+}{\partial x} = S, \quad (76)$$

where n_e and n_+ are the electron and ion densities, respectively, while j_e and j_+ denote the corresponding particle current densities. The source term S is given by:

$$S = |n_e \mu_e \mathcal{E}| \alpha_0 e^{-E_0/|\mathcal{E}|}, \quad (77)$$

where \mathcal{E} is the electric field, E_0 is the threshold field, μ_e the electron mobility and $\alpha_0 e^{-E_0/|\mathcal{E}|}$ is the ionization coefficient. To complete the model, the above equations need to be combined with the Poisson equation:

$$\frac{\partial \mathcal{E}}{\partial x} = \frac{e_0}{\epsilon_0} (n_+ - n_e), \quad (78)$$

and two phenomenological expressions for the particle current densities:

$$\begin{aligned} j_e &= -n_e \mu_e \mathcal{E} - D_e \frac{\partial n_e}{\partial x}, \\ j_+ &= 0, \end{aligned} \quad (79)$$

where e_0 is the absolute value of the electron charge, ϵ_0 is the permittivity of the vacuum and D_e is the electron diffusion coefficient.

Substituting $u = n_e$, $v = n_+$ and $\frac{\partial w}{\partial x} = \mathcal{E}$ and performing a normalization that sets most of the coefficients to 1, we end up with the following system of two PDEs and one ODE:

$$\begin{aligned} \frac{\partial u}{\partial t} &= u \left| \frac{\partial w}{\partial x} \right| e^{-\frac{1}{|\frac{\partial w}{\partial x}|}} - \frac{\partial}{\partial x} \left(u \frac{\partial w}{\partial x} - 0.1 \frac{\partial u}{\partial x} \right), \\ \frac{\partial v}{\partial t} &= u \left| \frac{\partial w}{\partial x} \right| e^{-\frac{1}{|\frac{\partial w}{\partial x}|}}, \\ \frac{\partial^2 w}{\partial x^2} &= u - v. \end{aligned} \quad (80)$$

with $x \in [0, 250]$ and $t \in [0, T_e]$. We use the following initial and boundary conditions:

$$\begin{aligned} u(x, 0) &= v(x, 0) = \frac{1}{100} e^{-(x-50)^2}, & w(x, 0) &= x, \\ u(0, t) &= u(250, t) = v(0, t) = v(250, t) = w(0, t) = 0.0, & \frac{\partial w}{\partial x}(250, t) &= 1.0. \end{aligned} \quad (81)$$

The third component equation is an ODE rather than a PDE. Furthermore, the associated solution variable w turns out to be several orders of magnitude larger than u and v . Consequently, if we let all three components contribute to the monitor function, the behaviour of w is likely to dominate the grid movement. Since we are mostly interested in the electron and ion densities u and v , we choose to set the w -contribution to the monitor function to zero.

Figure 27 shows the results for the electron density u for runs with different numbers of gridpoints N . We observe a rapid increase in u and expansion

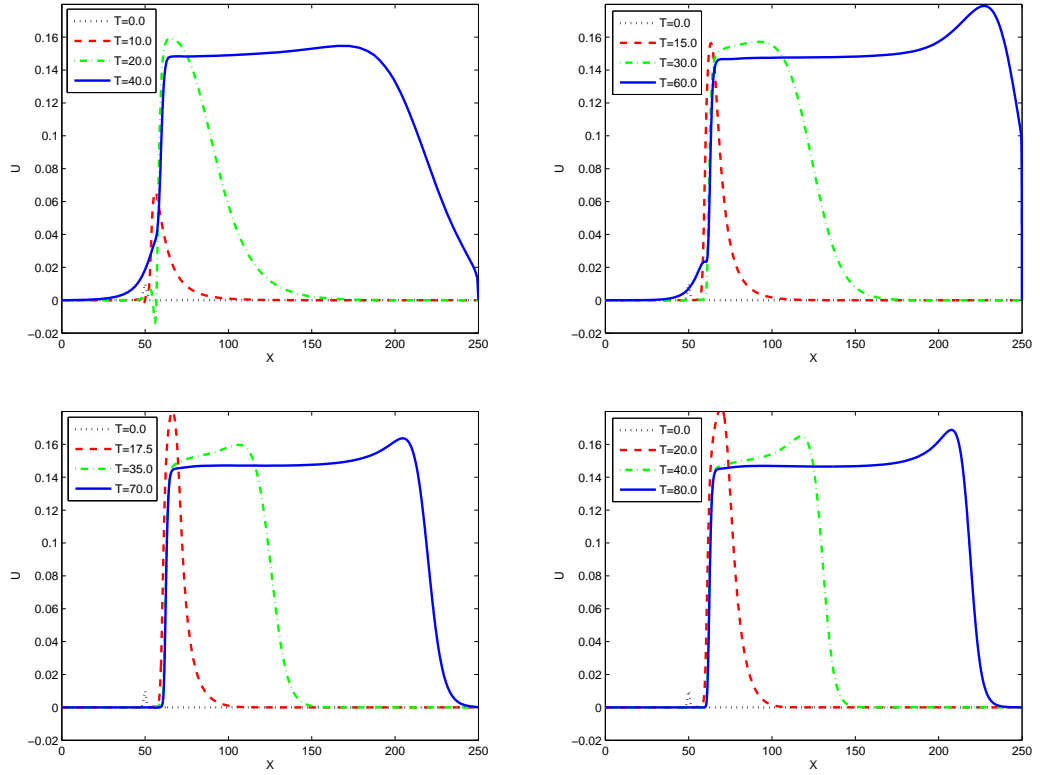


Figure 27: Solution $u(x, t)$ for the gas discharge problem: $N = 101$ (top left), $N = 201$ (top right), $N = 401$ (bottom left) and $N = 601$ (bottom right).

towards the right border, with a bump on the rightward-moving front. The ion density v behaves similarly, but with a bump on the left-hand side of the expanding cloud.

Note that T_e does not have the same value for the four different graphs in figure 27. The speed at which the front on u (the ionization front) moves to the right depends on the number of gridpoints N (related to the numerical accuracy). Consequently, the time at which the right border is reached also depends on N . Since the solver tends to crash shortly after the right border is reached, we were unable to complete a run with $T_e = 80.0$ for $N = 101, 201$, or 401 .

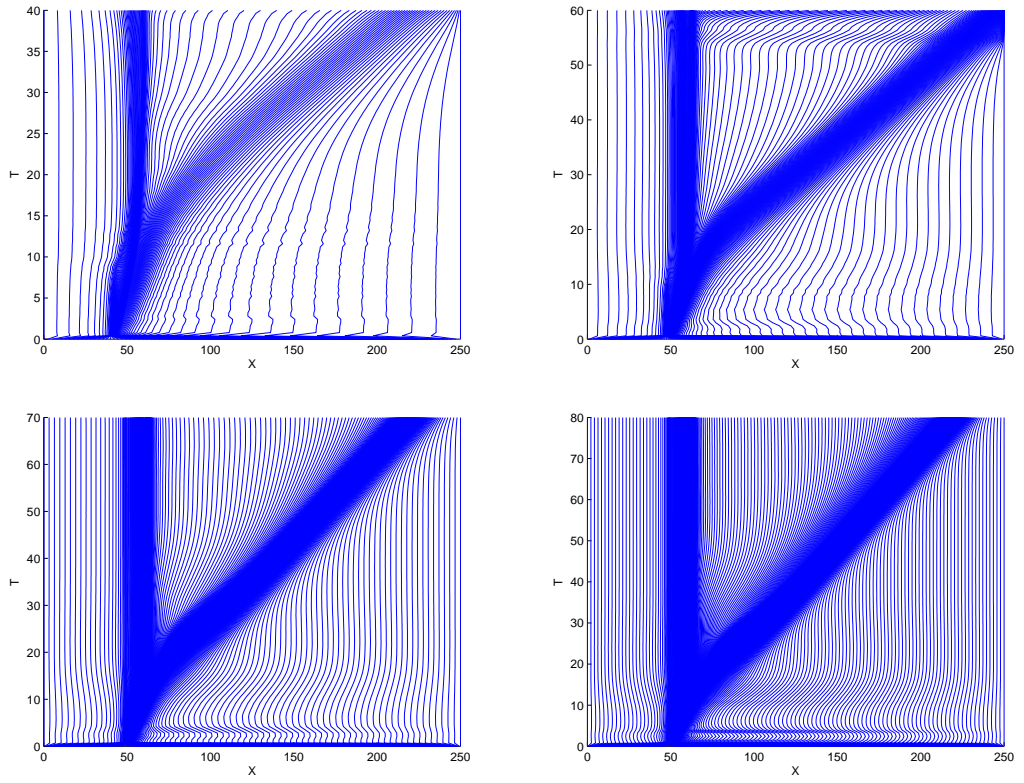


Figure 28: Grid movement for the gas discharge problem: $N = 101$ (top left), $N = 201$ (top right), $N = 401$ (bottom left) and $N = 601$ (bottom right).

Comparing the different results in figure 27, we note that the steepness of the rightward-moving front, as well as the speed at which it travels, is greatly affected by the number of gridpoints N , which is directly related to the numerical accuracy. Larger values of N (i.e. higher numerical accuracy), result in steeper, slower moving fronts. The travelling speed differences are more immediately obvious from figure 28, which shows the gridpoints following the moving front.

3.18 A Problem from Hydrology

In [17] and [18], we encounter the following equation as a model for Darcy flow in porous media with a dynamic capillary pressure relation:

$$\frac{\partial S}{\partial t} = \frac{\partial}{\partial x} \left\{ S^\alpha + S^\beta \frac{\partial S}{\partial x} + \varepsilon^2 S^\alpha \frac{\partial}{\partial x} \left(S^\gamma \frac{\partial S}{\partial t} \right) \right\} \quad (82)$$

We will consider a specific case of this equation with $\alpha = 2$ and $\beta = \gamma = 0$. The resulting equation contains a third order derivative $\frac{\partial^3 S}{\partial^2 x \partial t}$. Since we can only solve PDEs of the form (2) with the DASSL code, we introduce two new variables: $u = S$ and $v = \frac{\partial S}{\partial t}$ to convert this single PDE to a system of two PDEs that are of the required form (2):

$$\begin{aligned} \frac{\partial u}{\partial t} &= v, \\ \beta \frac{\partial v}{\partial t} + v &= \frac{\partial}{\partial x} \left[\frac{\partial u}{\partial x} + u^2 + \varepsilon^2 u^2 \frac{\partial v}{\partial x} \right]. \end{aligned} \quad (83)$$

We add the small term $\beta \frac{\partial v}{\partial t}$ with $0 < \beta \ll 1$, which does not appear in (82), to the second equation to make it time-dependent. We choose $\beta = 10^{-8}$. Since v is an artificial variable that allows us to solve this third order PDE (82), while u corresponds to the solution S of the original problem, we only take contributions from u for the monitor function that determines the grid movement.

We use the following initial conditions:

$$\begin{aligned} u(x, 0) &= 0, & \text{if } x < 0, \\ u(x, 0) &= 1, & \text{if } x \geq 0, \\ v(x, 0) &= 0, \end{aligned} \quad (84)$$

and boundary conditions:

$$u(-100, t) = 0, \quad u(100, t) = 1, \quad v(-100, t) = v(100, t) = 0, \quad (85)$$

with $x \in [-100, 100]$ and $t \in [0, T_e]$.

We solve this system for two different values of ε^2 . Figure 29 shows the results for $\varepsilon^2 = 0.4$ and figure 30 for $\varepsilon^2 = 2.0$. For a mathematical explanation why u is monotone for small and non-monotone for larger values of ε , we refer to [17].

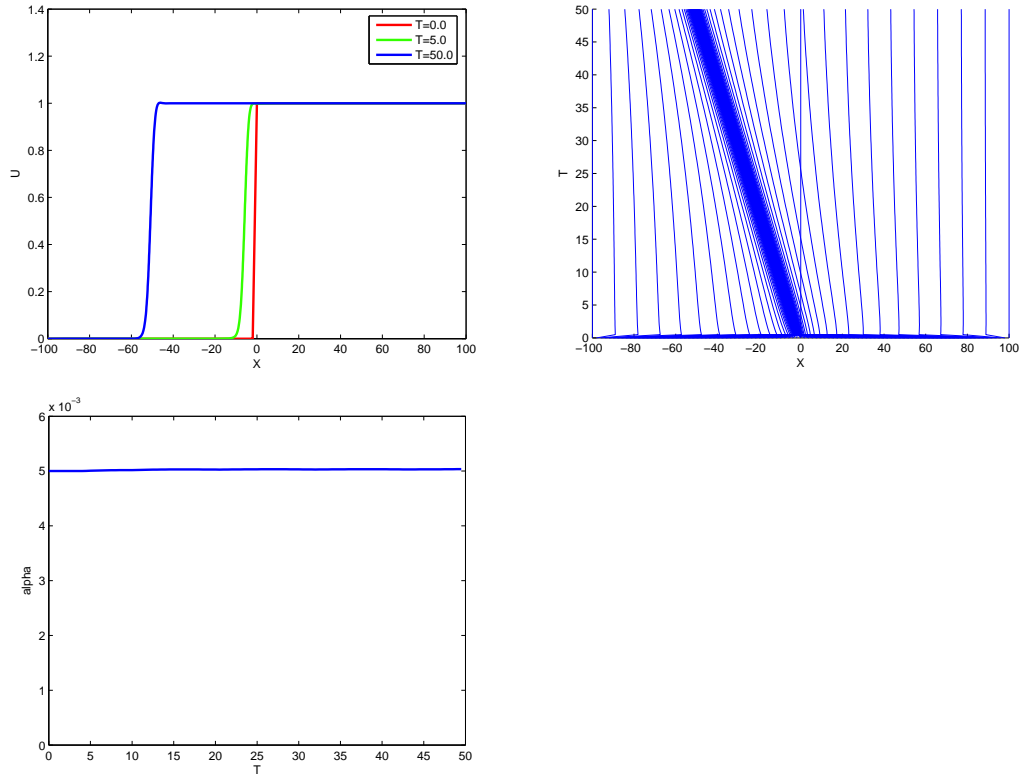


Figure 29: Results for the hydrology problem with $\varepsilon^2 = 0.4$: solution $u(x, t)$ (top left), grid movement $X_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 101$.

In figure 29, we see the steep slope on u travel leftward and become slightly smoother as it travels. The gridpoints follow this movement and $\alpha(t)$ is constant. The results for $\varepsilon^2 = 2.0$ shown in figure 30 are slightly different: a wave-like feature appears at the top of the leftward moving front. This results in a slightly wider area of high gridpoint density in the grid movement plot, as well as an increase in $\alpha(t)$ over time.

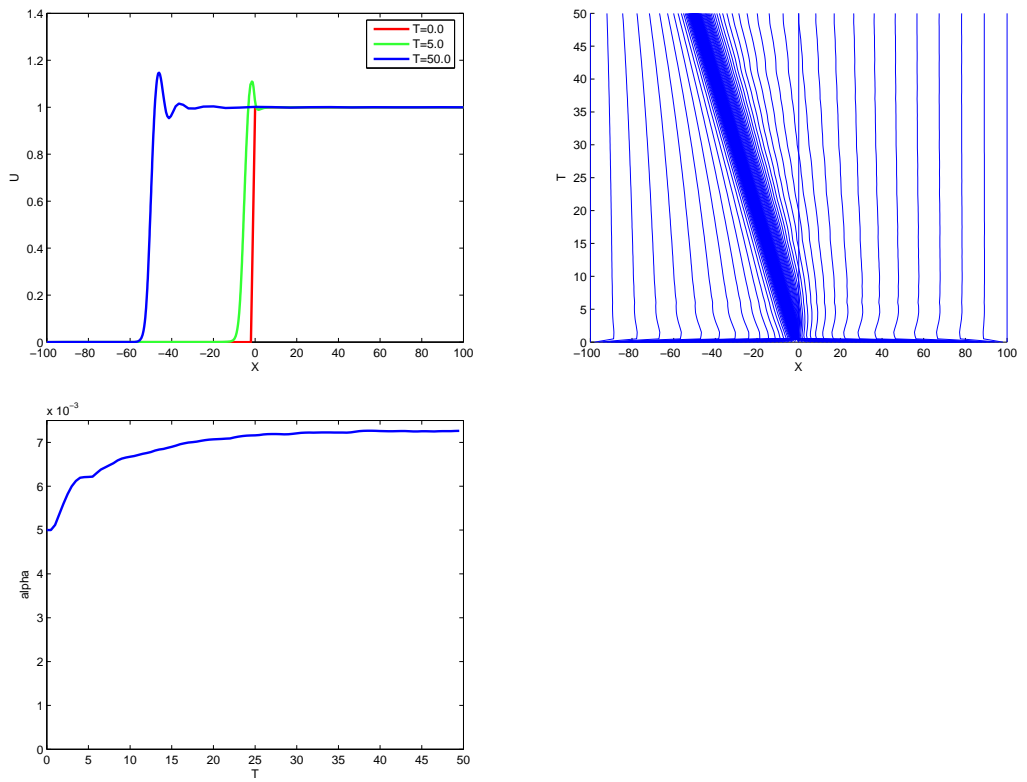


Figure 30: Results for the hydrology problem with $\varepsilon^2 = 2.0$: $u(x, t)$ (top left), grid movement $X_i(t)$ (top right) and $\alpha(t)$ (bottom left). $N = 101$.

3.19 A Model Describing the Formation of Liesegang Patterns

In [15], we encounter the following PDE to describe the formation of Liesegang patterns by the moving chemical front when an electrolyte A diffuses into a gel containing electrolyte B :

$$\frac{\partial m}{\partial t} = -\lambda \frac{\partial^2}{\partial x^2} \left(\varepsilon m - \gamma m^3 + \sigma \frac{\partial^2 m}{\partial x^2} \right) + S(x, t), \quad (86)$$

where m is the coarse grained magnetization, $S(x, t)$ is a space- and time-dependent source term, λ is a kinetic coefficient, ε measures the deviation from the critical temperature T_c ($\varepsilon > 0$ guarantees $T < T_c$), parameter $\gamma > 0$ ensures overall stability and $\sigma > 0$ provides stability against short-wavelength fluctuations.

Due to the higher order derivatives present in this model, it cannot be converted into a single PDE of the form (2), such that it can be solved by the code. However, the introduction of two new variables $u = m$ and $v = \frac{\partial^2 m}{\partial x^2}$ allows us to convert the single equation (86) into a system of two PDEs that are of type (2):

$$\begin{aligned} \frac{\partial u}{\partial t} &= -\lambda \varepsilon v + \lambda \frac{\partial^2}{\partial x^2} (\gamma u^3 - \sigma v) + S, \\ \beta * \frac{\partial v}{\partial t} &= \frac{\partial^2 u}{\partial x^2} - v. \end{aligned} \quad (87)$$

We use $\sigma = \frac{1}{400^2}$, $\gamma = \lambda = \varepsilon = 1.0$ and for the source term $S(x, t)$:

$$S(x, t) = \frac{0.181}{\sigma \left(\frac{t}{\sigma}\right)^{\frac{2}{3}}} \exp \left[\frac{1}{2} \left(\frac{400x - \sqrt{43.44 \left(\frac{t}{\sigma}\right)}}{4.54 \left(\frac{t}{\sigma}\right)^{\frac{1}{6}}} \right)^2 \right]. \quad (88)$$

We've added the small term $\beta * \frac{\partial v}{\partial t}$ with $0 < \beta \ll 1$, which does not appear in (86), to the second equation to make it time-dependent. We choose $\beta = 10^{-6}$. Since $v (= \frac{\partial^2 u}{\partial x^2}$ for $\beta = 0$) is an artificial variable that allows us to solve this third order PDE (86), similar to the problem in section 3.18, we only take contributions from u for the monitor function that determines the grid movement.

We use the following initial and boundary conditions, with $x \in [0, 1]$ and $t \in [0, T_e]$:

$$\begin{aligned} \frac{\partial u}{\partial x}(0, t) = \frac{\partial u}{\partial x}(1, t) = 0, & \quad v(0, t) = v(1, t) = 0, \\ u(x, 0) = -1, & \quad v(x, 0) = 0. \end{aligned} \quad (89)$$

The results for a run with $T_e = 0.013$ are shown in figure 31. Because of the small timescale, we've used a value of 10^{-4} for the temporal smoothing parameter τ in this experiment, instead of the default setting of $\tau = 10^{-3}$.

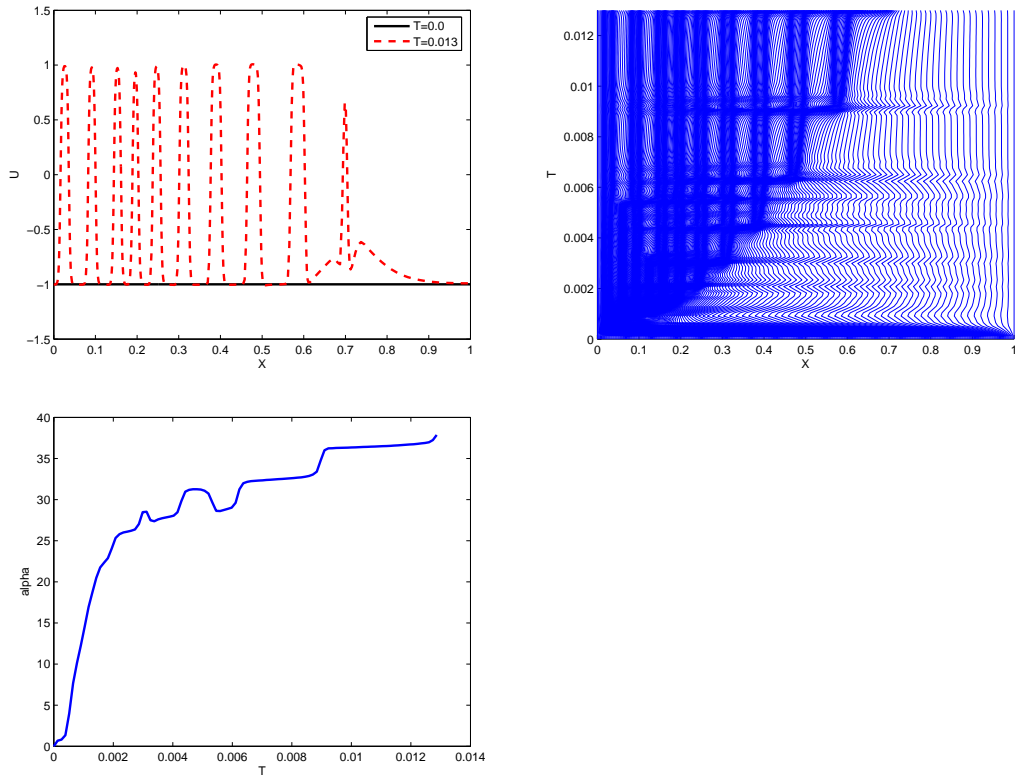


Figure 31: Results for the Liesegang pattern formation problem: $u(x, t)$ (top left), grid movement $X_i(t)$ (right) and $\alpha(t)$ (bottom left). $N = 401$, $\tau = 10^{-4}$.

Figure 31 shows how spikes on u appear, one by one, starting near the left boundary and more to the right as time progresses. The grid movement shows that each individual spike has a fixed position and the pattern expands to the right by adding new spikes. However, if only new spikes appear, $\alpha(t)$ should be monotonically increasing. The sudden decrease in $\alpha(t)$ around $t = 0.005$ suggests that something happens at that point in time that is not immediately clear from the two other graphs in figure 31. Upon closer inspection of $u(x, t)$, we find that a spike at $x = 0.06$ disappears around $t = 0.005$.

4 Possible Further Improvement

In the previous section, we have seen many PDE systems successfully solved with the moving grid method using monitor function (21), with time-dependent adaptivity parameter α given by (20). In this section, we will discuss a weakness of this monitor function by introducing a simple PDE system that highlights this weakness. We will also propose an improved monitor function to deal with such a system.

4.1 Three Implosions at Different Locations

Consider an "Implosion"-like problem with three independent component equations:

$$\frac{\partial u}{\partial t} = x \frac{\partial u}{\partial x}, \quad \frac{\partial v}{\partial t} = (x-1) \frac{\partial u}{\partial x}, \quad \frac{\partial w}{\partial t} = (x+1) \frac{\partial u}{\partial x}, \quad (90)$$

with $x \in [-3, 3]$, $t \in [0, T_e]$, and the initial and boundary equations such that the exact solutions $u_{ex}(x, t)$, $v_{ex}(x, t)$ and $w_{ex}(x, t)$ are, respectively:

$$u_{ex}(x, t) = e^{-(e^t x)^2}, \quad v_{ex}(x, t) = 10^{-2} e^{-(e^t(x-1))^2}, \quad w_{ex}(x, t) = 10^{-4} e^{-(e^t(x+1))^2}. \quad (91)$$

These exact solutions describe three Gaussian functions with peaks at different locations. They have different heights, though each should remain constant in height, while decreasing in width as time progresses.

The problem we encounter for this PDE system, when using monitor function (21), lies in the fact that the gradients of all three component solutions u , v and w are simply added up. Since the exact solutions for this system are several orders of magnitude apart, the u -term will dominate the monitor function and have a much larger influence on the grid movement than the v - and w -terms. Consequently, most gridpoints will concentrate around $x = 0$, leading to less accurate solutions around $x = 1$ and $x = -1$, where the peaks of v and w are. The results can be seen on the left-hand side of figure 32.

In the original monitor function:

$$M_i = (1 - \beta)\alpha(t) + \beta \sum_{k=1}^{NPDE} \left| \frac{\Delta U_i^k}{\Delta X_i} \right|, \quad i \in \{1, 2, \dots, N\}. \quad (92)$$

we have the following expression for α :

$$\alpha = \sum_{k=1}^{NPDE} \frac{\sum_{i=1}^N |\Delta U_i^k|}{x_R - x_L}. \quad (93)$$

We will now split up this α into $NPDE$ different parts α_k :

$$\alpha_k = \frac{\sum_{i=1}^N |\Delta U_i^k|}{x_R - x_L}, \quad (94)$$

and use each α_k only as a weight for its own component in the new monitor function:

$$M_i = (1 - \beta) + \beta \sum_{k=1}^{NPDE} \frac{1}{\alpha_k} \left| \frac{\Delta U_i^k}{\Delta X_i} \right|, \quad i \in \{1, 2, \dots, N\}. \quad (95)$$

We may encounter problems with this approach when $\alpha_k \approx 0$ for one or more components k , due to the factor $\frac{1}{\alpha_k}$ in (95). For $\alpha_k = 0$, we know we're dealing with a component that has a constant value for all x . Consequently, $\left| \frac{\Delta U_i^k}{\Delta X_i} \right|$ is zero for all $i \in \{1, 2, \dots, N\}$ and we can simply set $\alpha_k = 1$ (overwriting its previous value). However, for very small non-zero values, it can be hard to determine whether this is due to numerical inaccuracy or simply a legitimate solution on a very small scale. This makes it difficult to determine a 'threshold' value, below which we should treat α_k as if it were equal to zero (and consequently set its value to 1).

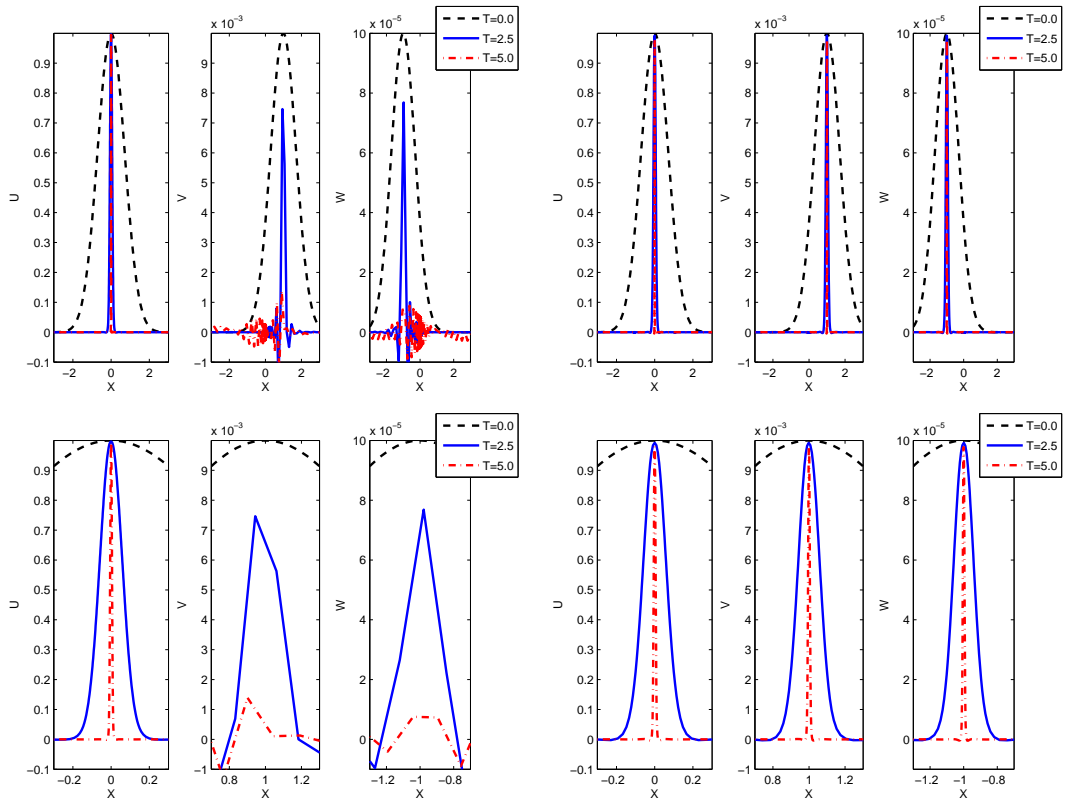


Figure 32: Results $u(x, t)$, $v(x, t)$ and $w(x, t)$ for the "three implosions" model: original monitor function (21) with $\alpha = \sum_k \alpha_k$ (top left), new monitor function (95) with separate α_k 's (top right), and zoomed in versions of each (bottom left and right). $N = 201$.

Figure 32 shows that the new monitor function indeed gives better results: all three components show a spike that becomes increasingly sharp, but maintains a constant height. The original monitor function manages to solve the large component $u(x, t)$ well, as was to be expected, but the smaller components $v(x, t)$ and $w(x, t)$ rapidly lose height and don't even have a Gaussian shape anymore at $t = T_e$.

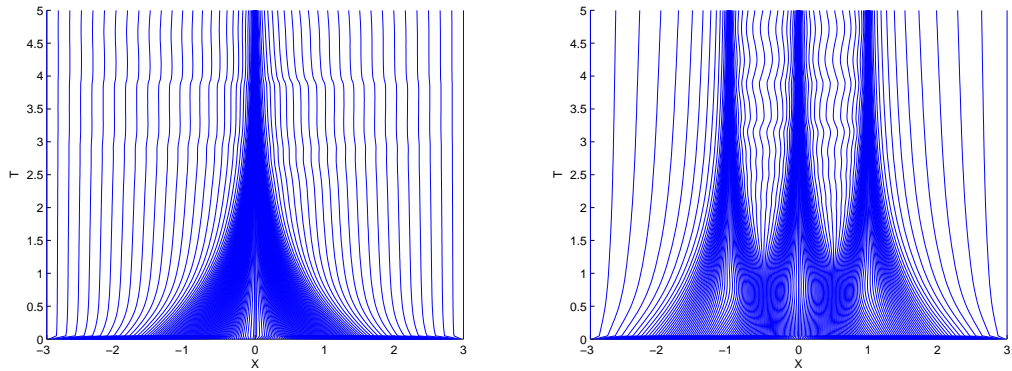


Figure 33: Grid movement for the "three implosions" model: original monitor function (21) with $\alpha = \sum_k \alpha_k$ (left) and new monitor function (95) with separate α_k 's (right). $N = 201$.

This difference in accuracy of the solutions of $v(x, t)$ and $w(x, t)$ between the two methods is illustrated by figure 33. With the new monitor function, gridpoints concentrate in the three separate areas associated with the three spikes (one spike for each component). With the original monitor function, the highest gridpoint concentration can be found in a narrow band around $x = 0$, where the spike on $u(x, t)$ is, while the gridpoint concentration is much lower outside of this band. Naturally, this lower gridpoint concentration at $x = -1$ and at $x = 1$, where the spikes on $v(x, t)$ and $w(x, t)$ are, results in less accurate solutions of the two components with smaller values.

4.2 Two Implosions at Different Locations and a Straight Line

We want to check whether the improved monitor function doesn't assign too much weight to solution components without large spatial gradients, such that it would result in a too uniformly distributed grid for PDEs with one very smooth component. To this end, we repeat the experiment from section 4.1, but with the second component equation replaced, such that its solution v is constant straight line:

$$\frac{\partial v}{\partial t} = 0, \quad (96)$$

with initial and boundary conditions determined by the exact solution $v_{ex} = 100x$. For u and w we keep the same equations and initial and boundary conditions as we had in section 4.1.

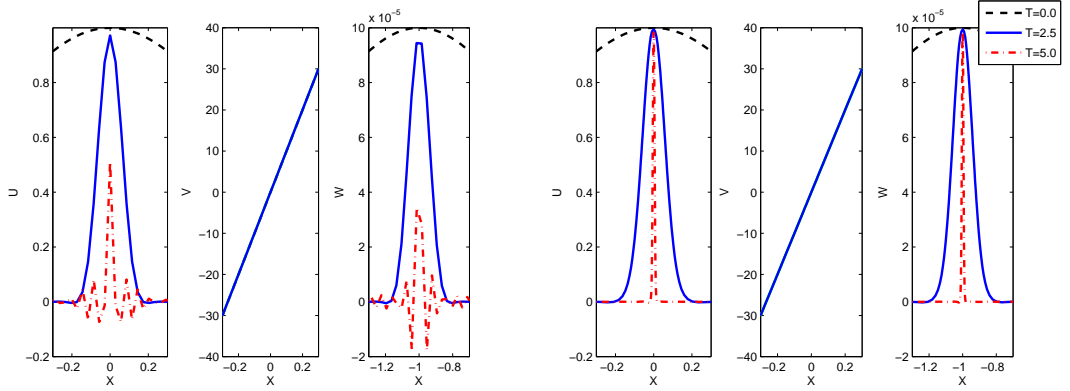


Figure 34: Results $u(x,t)$, $v(x,t)$ and $w(x,t)$ for the "two implosions" model: original monitor function (21) with $\alpha = \sum_k \alpha_k$ (left) and new monitor function (95) with separate α_k 's (right). $N = 201$.

With these choices, the straight line v has values that are several orders of magnitude larger than the two "implosion" components u and w . Consequently, v dominates the original monitor function. The results can be seen in figure 34: the solutions of u and w at time $t = T_e$, which should still have the same height they had at $t = 0.0$, have become much smaller. Figure 35 shows the cause: the gridpoint distribution with the original monitor function is relatively uniform. With the new monitor function, we achieve much better results: the gridpoints concentrate around the two spikes on u and w and those spikes come much closer to keeping a constant height.

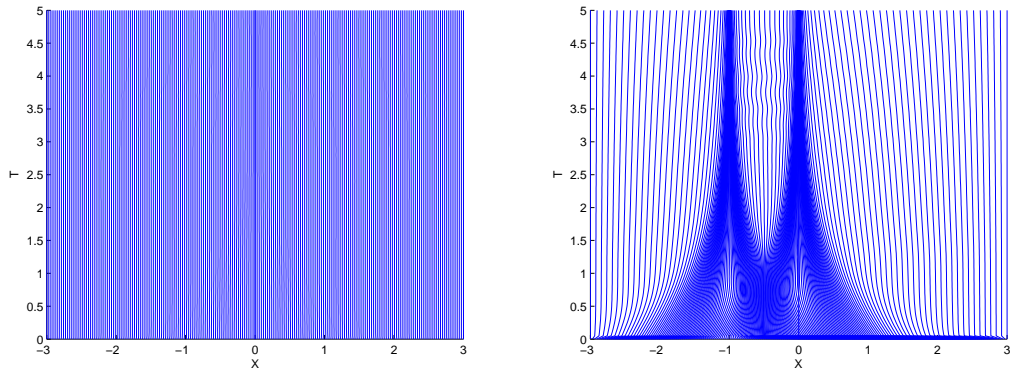


Figure 35: Grid movement for the "two implosions" model: original monitor function (21) with $\alpha = \sum_k \alpha_k$ (left) and new monitor function (95) with separate α_k 's (right). Note that we only show zoomed in versions. $N = 201$.

4.3 Gas Discharge with new monitor function

Finally, we try to solve the gas discharge problem from section 3.17, with the new monitor function. The results are shown in Figure 36. In section 3.17, we found that greater numerical accuracy (i.e. higher N) resulted in a slower-travelling front on u . With 401 gridpoints, the solution $u(x, t)$ we find now features a front that moves even slower than the solution we obtained using 601 gridpoints in section 3.17. It seems likely that this solution is the most accurate one. This suggests that the new monitor function may not only give better solutions for artificial problems like the three uncorrelated implosions, but also for more realistic problems like this gas discharge.

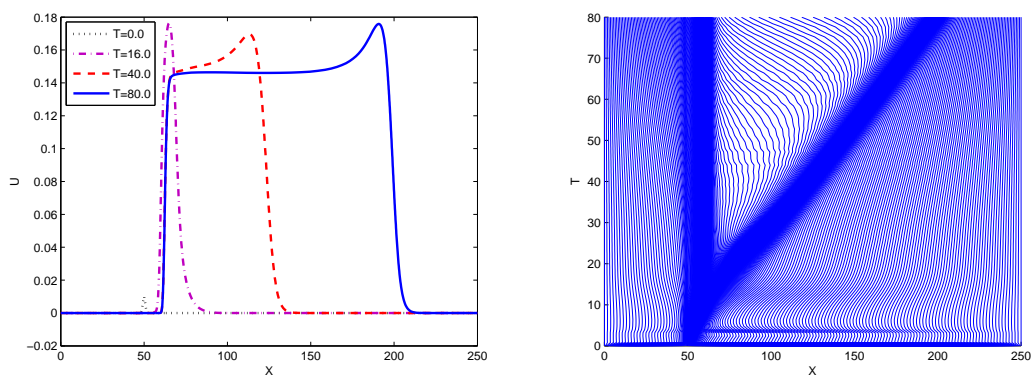


Figure 36: Results for the gas discharge model with new monitor function (95): $u(x, t)$ (left) and grid movement (right). $N = 401$.

5 Conclusions

We were able to use a moving grid method with time-dependent adaptivity parameter to successfully solve a large variety of test problems. Some of these problems differed greatly in scale of the spatial domain or solution values. The method has proven to be robust enough to solve such different problems without user input or problem re-scaling. Even those problems whose solutions change by orders of magnitude during the investigated timeframe. Only when we encounter problems with very short timescales do we require the user to either rescale the problem or choose a different temporal smoothing parameter.

We investigated the computational efficiency for one example: the implosion model. The additional computations, required to calculate the time-dependent adaptivity parameter in real time, result in a small efficiency loss compared to runs with a fixed adaptivity parameter α_f , if the user chooses a near-optimal value of α_f . However, this efficiency loss will generally be compensated by the time that is saved by not having to find this near-optimal value.

Not only does calculating $\alpha(t)$ in real-time make the moving grid method more robust, it can also provide a new method to visualize the way the solution changes over time. In several cases, a plot of the time-dependent adaptivity parameter $\alpha(t)$ provided us with information about the investigated problem, that was not as easily gathered from plots of the solution itself.

Finally, we've found that for some test problems consisting of more than one PDE, further improvements could be gained by calculating separate adaptivity parameters for each equation and using these to weigh the contribution of each component in the monitor function.

A Appendix: The Code

A.1 Overview

In our investigations, we have worked with a modified version of the code described in [5]. The programming language used is Fortran.

The complete code is contained in four separate files: *lib.f*, *spmdif.f*, *driver.f* and *prb*.f* (e.g. *prbGSpde.f* for the Gray-Scott problem). The main routine, from which all other routines are called, is contained in *driver.f*. Its basic structure is as follows:

1. Call problem initializer,
2. Initialize DASSL parameters,
3. Determine initial solution and initial grid,
4. Put semi-discrete PDE system + grid equation in DASSL,
5. Write grid and solution at specific steps (TPRINT) to outputfile.

The problem initializer routines are contained in *prb*.f*, while *spmdif.f* contains the moving-grid interface routines. The file *lib.f* contains all DASSL-related routines.

Ideally, a user wishing to apply this code to a new model should only be required to modify the *prb*.f* file, as described in [5]. However, the routines to write the results to the outputfile (*grid.dat*), as contained in the original *driver.f*, were written for a model consisting of exactly 3 PDEs. Consequently, to get meaningful output for a model with a different number of component equations, a user would have to modify these output-routines in *driver.f*. Since we only dealt with models consisting of 3 PDEs or less, we were able to work around this problem by adding one or two artificial component equations, as needed. To minimize the computation time spent on these artificial component equations, we chose the trivial case of a constant ($\frac{\partial u}{\partial t}(x, t) = 0$), uniform ($u(x, t) = 0$) function. Note that we made an exception for our computational efficiency measurements of the Implosion model, for which we did in fact modify the output-routines in *driver.f* and solved a true single-equation model.

A.2 Updates

Here we show the modifications we made to the code to change from a moving grid method with a fixed adaptivity parameter, to one with a time-dependent adaptivity parameter. We first made a few changes for ease-of-use. Originally, the user was prompted at the start of each run to manually input the values of several parameters (final time T_e , number of gridpoints N , temporal smoothing parameter τ and adaptivity parameter α). We choose to read these values (as well as spatial smoothing parameter κ) from a file, instead. This makes performing repeated runs with the same values for most or all of these parameters easier, faster and less prone to human error.

The first parameter, T_e , is read in *prb*.f*. We replace the following lines from the old *prb*.f*:

```
PRINT *, 'TE=?'  
READ *, TE
```

by the following in the new *prb*.f*:

```
C      Read settings from file:
      open (u, FILE='init.dat', STATUS='OLD')
      read(u,*) TE
      close (u)
```

to make the program read the file *init.dat* and assign the first value from the first line to T_e .

The other parameters are read in *driver.f*. In the original code, the user is asked to give the values of N and α , and to determine the value of τ , by choosing either a moving ($\tau = 10^{-6}$) or a fixed grid ($\tau = 10^8$).

```
      PRINT *, '# (moving) GRID POINTS =?'
      READ *, NPTS
      M      = NINT(DUMPRO(1))
*****      TAU      = 1E-3
*      PRINT *, 'TAU (= smoothing parameter in time-direction) =?'
*      READ *, TAU
      PRINT *, 'moving or fixed grid? (moving=1, fixed=0)'
32      READ *, HULP
      IF(HULP.EQ.1) THEN
          TAU=1.E-6
      ELSE IF(HULP.EQ.0) THEN
          TAU=1.E+8
      ELSE
          PRINT *, 'opnieuw keuze maken (1 of 0)'
          GOTO 32
      ENDIF
* KAPPA = 2.0 is standard
      KAPPA = 2.0
*** note: kappa plays the role of the parameter sigma...
*      PRINT *, 'SIGMA (= smoothing parameter in space-direction) =?'
*      READ *, KAPPA
* ALPHA = 0.01 is 'standard'
      ALPHA = 1.0D0
*      PRINT *, ' ALPHA (= adaptivity parameter in weight function) =?'
      READ *, ALPHA
```

Note that the code to ask the user for the value of κ (and the exact value of τ) is present in comment lines. We replace the above by the following:

```

C      Read settings from file:
      open (u, FILE='init.dat', STATUS='OLD')
      read(u,*) TE, NPTS, HULPTAU, ALPHA, HULPKAP
      close (u)
      PRINT *,'TE= ', TE
      PRINT *,'NPTS= ', NPTS
      PRINT *,'Moving grid= ', HULPTAU
      PRINT *,'ALPHA= ', ALPHA
      PRINT *,'KAPPA= ', HULPKAP

      IF(HULPTAU.EQ.1) THEN
        TAU=1.E-6
      ELSE IF(HULPTAU.EQ.0) THEN
        TAU=1.E+8
      ELSE
        TAU=HULPTAU
      ENDIF
      KAPPA = HULPKAP

```

Initially, we used the inputvalue HULPTAU as a boolean to switch the moving grid on (1) or off (0), just as it was used in the original code. However, for PDEs with either very long or very short time-scales, the standard values of τ for "on" (10^{-6}) or off (10^8) may not have the desired effect. To deal with such cases, we added the option to give τ any value (other than 0 or 1), simply by assigning the value of HULPTAU to τ if it's neither 0 nor 1.

The second and more important set of changes we made to the code deal with the calculation of the time-dependent adaptivity parameter $\alpha(t)$. We initially added the code for this calculation to *spmdif.f*, which is where α is used, since it contains the routine that calculates the monitor function. However, it turned out that re-calculating $\alpha(t)$ for every time the monitor function is calculated made the program extremely slow. We decided to calculate $\alpha(t)$ at fixed time-intervals.

In the original code, the results are written to file at NPRINT time-intervals. These intervals (we used NPRINT = 100 for all our runs) are determined in *prb*.f*:

```

      NPRINT = 100
      DO 100 I=1,NPRINT
        TPRINT(I) = REAL(I)*TE/NPRINT
      100 CONTINUE

```

In the original *driver.f*, the DASSL routines, followed by commands to write the results to file, are called in a loop going over these intervals:

```

      open(unit=16,file='grid.dat')
c      write grid points to file
      do 88 i=1,npts
        write(16,44) i,k,y(4*i),y(4*i-3),y(4*i-2),y(4*i-1)
88      continue

      DO 10 IPRINT = 1, NPRINT
        TOUT = TPRINT(IPRINT)
15      CALL DDASSL (RESID, NEQ, T, Y, YPRIME, TOUT, INFO, RTOL, ATOL,
+                IDID, RWORK, LRW, IWORK, LIW, RWK, IPAR, JAC)

      do 451 i=1,npts
        write(16,44) i,IPRINT,y(4*i),y(4*i-3),y(4*i-2),y(4*i-1)
451      continue

```

In our modified *driver.f*, the commands to calculate $\alpha(t)$ are inserted into this loop before the call to the DASSL routines:

```

      NPDE1=NPDE+1
C      y(4*i),y(4*i-3),y(4*i-2),y(4*i-1)
C      = X      U      V      W
      open(unit=37,file='alfa_uit2.dat')
      DO 10 IPRINT = 1, NPRINT
        TOUT = TPRINT(IPRINT)
        ALFADYN = 0.0
        DO 13 KB = 1, 2
          ALFADYNK(KB) = 0.0
          DO 23 IB = 1, NPTS-1
C           DU = Y(K,I+1)-Y(K,I)
C           DX = Y(4*(IB+1))-Y(4*IB)
C           SQRUX = ABS(DU/DX)
C           SQRUX(KB) = ABS((Y(KB-4+4*(IB+1))-Y(KB-4+4*IB))/DX)
          ALFADYNK(KB) = ALFADYNK(KB) + (SQRUX(KB)*DX)
23          CONTINUE
          ALFADYN = ALFADYN + ALFADYNK(KB)
13          CONTINUE
          DYNALF=ALFADYN/Y(4*(NPTS))-Y(4)
          write(37,*) DYNALF, T
          IF(IPRINT .EQ. 1) THEN
            PRINT *, 'First DYNALF= ', DYNALF
          ENDIF

15      CALL DDASSL (RESID, NEQ, T, Y, YPRIME, TOUT, INFO, RTOL, ATOL,
+                IDID, RWORK, LRW, IWORK, LIW, RWK, IPAR, JAC)

```

Note that the way $\alpha(t)$ is calculated assumes a model with 3 component PDEs. We've added comments to the code to clarify how these 3 components + the moving grid are stored in the solution vector Y. To make the value of $\alpha(t)$ calculated here available in the other routines in *driver.f*, we add the following lines to the start of each routine:

```

      DOUBLE PRECISION DYNALF
      COMMON /COMDYNALF/ DYNALF

```

To make the value of $\alpha(t)$ available to the routine in *spmdif.f* where the monitor function is calculated, we added an argument (DYNALF) to the relevant routine (SKMRES), by changing the following line from *spmdif.f*:

```

      SUBROUTINE SKMRES (NEQN, T, Y, YDOT, RES, IRES, RWK, NRWK)

```


to:

```
SUBROUTINE SKMRES (NEQ, T, Y, YPRIME, DELTA, IRES, RWK, NRWK, DYNALF)
```

Naturally, the line in *driver.f* which calls this routine is modified accordingly. Since the monitor function is calculated in a different subroutine (XMNTR) of *sprmdif.f*, which is called from SKMRES, we added the following line to SKMRES:

```
DYNALF2 = DYNALF
```

and made this variable (DYNALF2) available to XMNTR by adding the following lines to both routines:

```
DOUBLE PRECISION DYNALF2
COMMON /COMDYNALF2/ DYNALF2
```

Finally, we modified the calculation of the monitor function in this subroutine XMNTR of *sprmdif.f*. Note that the time-dependent adaptivity parameter was not the only difference between the monitor function used in the original code:

$$M_i = \sqrt{\alpha + \sum_{j=1}^{N_{PDE}} \left| \frac{\Delta U_i^j}{\Delta X_i} \right|^2}, i \in \{1, 2, \dots, N\}, \quad (97)$$

and the monitor function we used:

$$M_i = (1 - \beta)\alpha(t) + \beta \sum_{j=1}^{N_{PDE}} \left| \frac{\Delta U_i^j}{\Delta X_i} \right|, i \in \{1, 2, \dots, N\}. \quad (98)$$

We chose $\beta = 0.8$ for all our experiments.

The original code contained the following subroutine XMNTR (we have removed some comments for brevity):

```

SUBROUTINE XMNTR (Y, G, NPDE, N)
INTEGER NPDE, N
DOUBLE PRECISION Y(NPDE+1,0:N+1), G(NPDE+1,0:N+1)
C
C-----
C Purpose:
C -----
C Compute monitor for grid equation,
C   M_i = M(x(i+1/2)) = sqrt(alfa + !!ux!!**2)
C
C Exit:
C   G(NPDE+1,i) = M(i)
C-----
DOUBLE PRECISION TAU, RKAPPA, ALFA
COMMON /METPAR/ TAU, RKAPPA, ALFA
SAVE /METPAR/
C-----
INTEGER I, K, NPDE1
DOUBLE PRECISION DU, DX, SUX2
NPDE1 = NPDE+1

DO 10 I = 0, N
  SUX2 = 0.0
  DO 20 K = 1, NPDE
    DU = Y(K,I+1)-Y(K,I)
    SUX2 = SUX2 + DU*DU
  20 CONTINUE
  DX = Y(NPDE1,I+1)-Y(NPDE1,I)
  SUX2 = SUX2 / (DX*DX)
  G(NPDE1,I) = SQRT(ALFA + SUX2/NPDE)
10 CONTINUE
RETURN
END

```

Our modified XMNTR looked as follows:

```

SUBROUTINE XMNTR (Y, G, NPDE, N)
INTEGER NPDE, N
DOUBLE PRECISION Y(NPDE+1,0:N+1), G(NPDE+1,0:N+1)
C-----
C Purpose:
C Compute monitor for grid equation, M(i)
C Exit:
C G(NPDE+1,i) = M(i)
C-----
DOUBLE PRECISION TAU, RKAPPA, ALFA
DOUBLE PRECISION DYNALF2
COMMON /COMDYNALF2/ DYNALF2
COMMON /METPAR/ TAU, RKAPPA, ALFA
SAVE /METPAR/
C-----
INTEGER I, K, NPDE1
DOUBLE PRECISION DU, DX, SUX2, SQRUX(2), ALFAK(2)
NPDE1 = NPDE+1

ALFA=DYNALF2
DO 12 I = 0, N
IF (ALFA .EQ. 0) THEN
G(NPDE1,I) = 1.0
ELSE
DO 22 K = 1, 2
DX = Y(NPDE1,I+1)-Y(NPDE1,I)
SQRUX(K) = ABS((Y(K,I+1)-Y(K,I))/DX)
22 CONTINUE
C Beta=0.8
G(NPDE1,I) = (0.2*ALFA + 0.8*SQRUX(1)+ 0.8*SQRUX(2))
ENDIF
12 CONTINUE
RETURN
END

```

References

- [1] Van Dam, A. and Zegeling, P. A. **2006** A Robust Moving Mesh Finite Volume Method Applied to 1D Hyperbolic Conservation Laws from Magnetohydrodynamics. *Journal of Computational Physics* V216, N2 526-546.
- [2] Petzold, L. R. **1983** A Description of DASSL: A Differential/Algebraic System Solver. In: R. S. Stepleman, ed., *IMACS Transactions on Scientific Computation*, North-Holland, Amsterdam.
- [3] Zegeling, P. A. **2005** On Resistive MHD Models with Adaptive Moving Meshes. *J. on Scientific Computing* 24, 263-284.
- [4] Mackenzie, J. A. and Mekwi, W. R. **2007** On The Use of Moving Mesh Methods to Solve PDEs. Pages 243-278 in: Tao Tang, Jinchao Xu., eds., *Adaptive Computations: Theory and Algorithms. Mathematics Monograph Series 6*, Science Press, Beijing.
- [5] Blom, J. G. and Zegeling, P. A. **1994** Algorithm 731: A Moving Grid Interface for Systems of One-Dimensional Time-Dependent Partial Differential Equations. *ACM Trans. Math. Softw.* 20, 2(June), 194-214.
- [6] Verwer, J. G., Blom, J. G., Furzeland, R. M. and Zegeling, P. A. **1989** A Moving-Grid Method for One-Dimensional PDEs Based on the Method of Lines. In: J. E. Flaherty, P. J. Paslow, M. S. Shepherd and J. D. Vasilakis, eds., *Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia.
- [7] Zegeling, P. A., Blom, J. G. **1992** An Evaluation of the Gradient-Weighted Moving-Finite-Element Method in One Space Dimension. *Journal of Computational Physics* V103, N2 422-441.
- [8] Skeel, R. D. and Berzins, M. **1990** A method for the spatial discretization of parabolic equations in one space variable. *SIAM J. Sci. Stat. Comput.* 11, 1(Jan.), 1-32.
- [9] Zegeling, P. A., Verwer, J. G. and Van Eijkeren, J. C. H. **1992** Application of a Moving Grid Method to a Class of 1D Brine Transport Problems in Porous Media. *Int. J. Numer. Methods Fluids* 15, 2(July), 175-191.
- [10] Doelman, A., Kaper, T. J., Zegeling, P. A. **1997** Pattern Formation in the 1-D Gray-Scott Model. *Nonlinearity* 10, 523-563.
- [11] Zegeling, P. A., Kok, H. P. **2002** Adaptive Moving Mesh Computations for Reaction-Diffusion Systems. *J. of Comp. and Appl. Maths* 168, 519-528.
- [12] Furzeland, R. M., Verwer, J. G. and Zegeling, P. A. **1990** A Numerical Study of Three Moving Grid Methods for One-Dimensional Partial Differential Equations Which Are Based on the Method of Lines. *Journal of Computational Physics* V89, 349-388.
- [13] Zegeling, P. A. **2007** Theory and Applications of Adaptive Moving Grid Methods. Chapter 7 in: Tao Tang, Jinchao Xu., eds., *Adaptive Computations: Theory and Algorithms. Mathematics Monograph Series 6*, Science Press, Beijing.

- [14] Pieters, G. J. M. **2004** Stability and Evolution of Gravity-Driven Flow in Porous Media. *Ph. D. Thesis, TU Eindhoven, ISBN 90-386-0932-9.*
- [15] Antal, T., Droz, M., Magnin, J., and Rácz, Z. **1999** Formation of Liesegang Patterns: A Spinodal Decomposition Scenario. *Phys. Rev. Lett.* *83*, 2880 - 2883.
- [16] Ebert, U., Van Saarloos, W. and Caroli, C. **1997** Propagation and Structure of Planar Streamer Fronts. *Phys. Rev. E* *55*, 1530 - 1549.
- [17] Cuesta, C., Hulshof, J. **2001** A Model Problem for Unsaturated Porous Media Flow with Dynamic Capillary Pressure. *CWI Report MAS-R0104.*
- [18] Cuesta, C., Van Duijn, C. J., and Hulshof, J. **2000** Infiltration in porous media with dynamic capillary pressure: travelling waves. *Euro. Journal of Applied Math.* *11*, 381-397.
- [19] Van den Berg, J. B., King, J. R., and Hulshof, J. **2003** Formal Asymptotics of Bubbling in the Harmonic Map Heat Flow. *SIAM Journal on Applied Mathematics* *63*, 1682-1717.
- [20] Chaplain, M. A. J., and Stuart, A. M. **1993** A Model Mechanism for the Chemotactic Response of Endothelial Cells to Tumour Angiogenesis Factor. *Mathematical Medicine and Biology* *10(3)*, 149-168.
- [21] Planetmath: FitzHugh Nagumo Equation <http://planetmath.org/encyclopedia/FitzHughNagumoEquation.html>, 3 September 2007.