Dynamic iteration for time-dependent partial differential equations: A first approach

Morten Bjørhus
Department of Mathematical Sciences
The Norwegian Institute of Technology
N-7034 Trondheim, Norway *

September 16, 1992

Abstract

In this paper we consider the use of parallel computing in the context of solving systems of time-dependent PDEs. After a review of dynamic iteration for systems of ODEs, we generalize the concept of such methods to the PDE case through the use of domain decomposition. The result is an iterative scheme (ODI) in which the original problem is transformed into several problems on the subdomains. These problems are completely independent of each other in every iteration, and this makes the technique suitable for parallel processing. Convergence results are given for the linear hyperbolic problem with constant coefficients in one space dimension. We discuss the ODE formulation for the systems in the hyperbolic ODI case, followed by some numerical experiments.

Mathematics Subject Classification (1991): 65M99, 65Y05, 65L05

1 Introduction

In numerous applications one is faced with the problem of solving systems of time-dependent differential equations. The solution of the system is often approximated by some numerical method, and a variety of such methods is available for most problems. As a consequence of the increased use of parallel computers seen in the last decade, there has been a growing interest for numerical algorithms designed with these special architectures in mind. For ordinary differential equations (ODEs), a technique that seems particularly interesting for parallel computers is the dynamic iteration, or waveform relaxation methods. As the name indicates, dynamic iteration techniques are solution techniques for differential equations in which the iterates are functions; not merely numbers or vectors as we know from many other iterative schemes, like for the solution of linear algebraic systems. As an example of a dynamic iteration method for the solution of the nonlinear autonomous initial value problem

$$\dot{u}_1 = f_1(u_1, u_2)\,, \tag{1}$$

$$\dot{u}_2 = f_2(u_1, u_2)\,, \tag{2}$$

*e-mail: mortenb@imf.unit.no

$$u_1(0) = u_{10} \in \mathbb{R}\,, \tag{3}$$
$$u_2(0) = u_{20} \in \mathbb{R}\,, \tag{4}$$

where $u_1, u_2 \in C([0,T];\mathbb{R})$ and $f_i : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, we may consider the iterative scheme

$$\dot{u}_1^{(k+1)} = f_1(u_1^{(k+1)},\ u_2^{(k)}\ )\,, \tag{5}$$
$$\dot{u}_2^{(k+1)} = f_2(\ u_1^{(k)}\ ,u_2^{(k+1)})\,, \tag{6}$$

$$u_1^{(k+1)}(0) = u_{10}\,, \tag{7}$$
$$u_2^{(k+1)}(0) = u_{20}\,; \tag{8}$$

in which $k = 0,1,2,\ldots$. For the initial iterates, we may choose e.g. $u_1^{(0)}(t) \equiv u_{10}$ and $u_2^{(0)}(t) \equiv u_{20}$. We see that the system (5)-(8) is decoupled; i.e. the equations for $u_1^{(k+1)}$ and $u_2^{(k+1)}$ are independent. We have thus reduced the problem down to solving two single decoupled differential equations, but we have to pay for introducing this artificial decoupling by performing several iterations, or *sweeps*, over the interval $[0,T]$. It is shown below that the sequence of iterates $\{u_1^{(k)}\}$ and $\{u_2^{(k)}\}$ converges linearly and uniformly to the solution u_1 and u_2 of (1)-(4) on every finite interval $[0,T]$; whenever f_1 and f_2 are Lipschitz continuous. The kind of decoupling introduced to the system is referred to as a *Jacobi splitting*, while the decoupling we get by replacing $u_1^{(k)}$ in (6) by $u_1^{(k+1)}$, is called a *Gauss-Seidel splitting*. The analogy to the language of iterative linear solvers is immediate. We note that since the Jacobi splitting splits the problem into two independent problems in each iteration, we can solve these in parallel if we have two processors available. In Section 2 we will look at a general definition of dynamic iteration for ODEs, together with a brief presentation of some results.

The philosophy behind the dynamic iteration techniques is to impose an artificial decoupling to the problem. For all practical purposes these are dependency structures that decouple some or all of the equations, in order to make the resulting systems smaller and independent. Hence they can be integrated in parallel, and each subsystem requires less memory; the latter is especially important when using implicit integration schemes. Now, switching to the PDE context, it is natural to ask if it is possible to use a similar approach there. By using the fact that we through domain decomposition can obtain several smaller problems on the subdomains, we propose in Section 3 a general dynamic iteration approach for time-dependent PDEs, called Outer Dynamic Iteration (ODI). The idea of the ODI algorithm is closely related to the ideas of subdomain iteration as proposed by Quarteroni [18]; on the other hand the ODI idea is quite different, as it does not assume any kind of discretization of the system. We also briefly discuss possibilities for the application of dynamic iteration on the resulting discretized ODE system; in that case we refer to the methods as Inner Dynamic Iteration (IDI).

In Section 4 we present a specific ODI algorithm for quasilinear hyperbolic systems, and in Section 5 we prove that it converges for linear problems in one space dimension. In Section 6 we show how to formulate the discretized hyperbolic PDE as a well defined ODE system, in order to be able to apply an arbitrary time stepping scheme to it. Then, in Section 7, we perform some numerical experiments to validate the applicability of the proposed ODI algorithm for the hyperbolic systems. Both linear and quasilinear systems are solved.

2 Dynamic iteration for ODEs

2.1 Definitions, examples

In this section, we will review some of the basic concepts and results concerning dynamic iteration as a means for solving ODE initial problems. The idea of utilizing a scheme like the one in (5)-(8) in order to find a solution of (1)-(4) is not new. It was used by both Picard [16] and Lindelöf [11] in the last decade of the 19th century, as a tool to show the existence and uniqueness of solutions of ODE initial value problems. However, the technique did not get much attention as a practical method until Lelarasmee *et al.* [9] published a paper on its application in the process of simulating VLSI circuits. In their paper, the method was called the *Waveform Relaxation* method, and the publication of this paper initiated the recent research activity on such methods for the solution of general ODE initial value problems. Miekkala and Nevanlinna [12] named the technique *Dynamic Iteration* when used on linear ODEs, while Nevanlinna [13][14][15] has recently referred to the technique as *Picard-Lindelöf Iteration*. In this paper, we have chosen to refer to these techniques, together with their generalizations in the context of time-dependent PDEs, as *Dynamic Iteration techniques*.

The motivation for the recent study of dynamic iteration for ODEs, is the demand for more efficient numerical integration procedures for large systems of ODEs. Such systems often appear in applications, e.g. in VLSI circuit simulation and in fluid dynamics after spatial discretization; and when the size of the system increases, the system of ODEs is likely to get stiff. Hence, stable implicit integration schemes are preferred (cf. e.g.[5]), in order to avoid the step size to be governed by the lack of stability rather than by the demand for accuracy. However, implicit methods are very expensive computationally, since they involve the solution of a nonlinear system of equations in every stage. In practical codes, the nonlinear solver is some variant of the Newton iterative method, involving the solution of a linear system in each iteration which is (generally) of complexity $\mathcal{O}(N^3)$ where N is the size of the system. The idea of dynamic iteration is to introduce an artificial decoupling, or splitting, to the ODE system; such that it is decoupled into several subsystems of smaller size. Hence, each subsystem can be integrated independently, with the step size adjusted to the dynamics within it. This leads to a more efficient time integration for the slowly varying subsystems, as these do not have to consider the more rapid change in other subsystems. Maybe more important is the possibility of doing the integration in all subsystems concurrently, given a parallel computer environment. However, the artificial decoupling introduces an error in the computed solution which we try to eliminate by performing several sweeps over the time interval of interest; at each sweep taking care of the coupling between the different subsystems by reading the values of the coupling variables computed in the previous sweep. Taking into account the possibility of obtaining more efficient time integration in each subsystem and the possibility for concurrency, the dynamic iteration technique is an interesting alternative for large systems of ODEs. It seems especially applicable in situations where the system already has a modular structure with weak coupling between the subsystems. We have the following definition.

DEFINITION 2.1 (DYNAMIC ITERATION FOR ODE) *Given the ODE initial value problem*

$$\dot{u} = f(u) \; , \tag{9}$$

$$u(0) = u_0 \; , \tag{10}$$

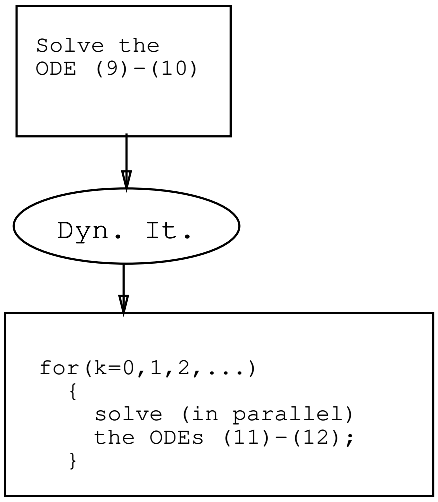

Figure 1: The Dynamic Iteration technique. If we choose a disjoint splitting $\tilde{f}$, the ODEs can be solved concurrently

where $u : [0,T] \to \mathbb{R}^m$ *and* $f : \mathbb{R}^m \to \mathbb{R}^m$, *we call the following iterative scheme a dynamic iteration scheme for (9)-(10):*

$$\dot{u}^{(k+1)} = \tilde{f}(u^{(k+1)}, u^{(k)}) \ , \tag{11}$$

$$u^{(k+1)}(0) = u_0 \ , \tag{12}$$

$k = 0, 1, 2, \ldots$ *; where* $\tilde{f} : (v, w) \in \mathbb{R}^m \times \mathbb{R}^m \mapsto \tilde{f}(v, w) \in \mathbb{R}^m$ *must satisfy the following condition:*

$$\tilde{f}(v, w) \mid_{\substack{v=u \\ w=u}} = f(u) \ . \tag{13}$$

The initial iterate $u^{(0)}(t)$ *can be chosen freely, but must satisfy the initial condition (12).*

REMARK 2.2 We note that the following holds (when f is differentiable):

$$\frac{\partial \tilde{f}}{\partial v} \mid_{\substack{v=u \\ w=u}} + \frac{\partial \tilde{f}}{\partial w} \mid_{\substack{v=u \\ w=u}} = \frac{\partial f}{\partial u} \mid_u \ ; \tag{14}$$

i.e. we can view the dynamic iteration approach as a method for keeping only a selected part of the original problem's dependency structure active, by taking care of the remainder through the previous iterate. This can be shown by applying the definition of the derivative to the right hand side of (14).

One should note that this definition of a dynamic iteration method is quite general, and contains e.g. the definition of waveform relaxation found in [23] and the dynamic iteration methods studied by Miekkala and Nevanlinna in [12]. Let us in the example given in the introduction choose f_1 and f_2 to be the two nonlinear functions

$$f_1(u) = u_1 + u_2 + 2u_1(\sin u_2)^2 \ , \tag{15}$$

$$f_2(u) = -u_1 + \sin u_2 \ . \tag{16}$$

A Jacobi splitting scheme for (1)-(2) would then be given by

$$
\begin{aligned}
\dot{u}_1^{(k+1)} &= u_1^{(k+1)} + 2u_1^{(k+1)}(\sin u_2^{(k)})^2 + u_2^{(k)} \ , && (17) \\
\dot{u}_2^{(k+1)} &= -u_1^{(k)} + \sin u_2^{(k+1)} \ . && (18)
\end{aligned}
$$

In the language of the above definition, we here have that

$$
\tilde{f}(v,w) = \left(\begin{array}{c} v_1 + 2v_1(\sin w_2)^2 + w_2 \\ -w_1 + \sin v_2 \end{array} \right) \ . \tag{19}
$$

It is clear that (13) is fulfilled, and thus the scheme is a dynamic iteration scheme according to the definition. We also see that

$$
\frac{\partial \tilde{f}}{\partial v} \mid_{\substack{v=u \\ w=u}} = \left[\begin{array}{cc} 1 + 2(\sin u_2)^2 & 0 \\ 0 & \cos u_2 \end{array} \right] \ , \ \frac{\partial \tilde{f}}{\partial w} \mid_{\substack{v=u \\ w=u}} = \left[\begin{array}{cc} 0 & 4u_1 \sin u_2 \cos u_2 + 1 \\ -1 & 0 \end{array} \right] \ , \tag{20}
$$

and

$$
\frac{\partial f}{\partial u} \mid_u = \left[\begin{array}{cc} 1 + 2(\sin u_2)^2 & 4u_1 \sin u_2 \cos u_2 + 1 \\ -1 & \cos u_2 \end{array} \right] \ , \tag{21}
$$

which confirms the relation (14).

The definition allows for very general splittings. Consider e.g. the splitting of the same system given by

$$
\begin{aligned}
\dot{u}_1^{(k+1)} &= u_1^{(k+1)} + u_2^{(k+1)} + 2u_1^{(k)}(\sin u_2^{(k)})^2 \ , && (22) \\
\dot{u}_2^{(k+1)} &= -u_1^{(k+1)} + \sin u_2^{(k)} \ . && (23)
\end{aligned}
$$

This would also be a dynamic iteration scheme, and we would in this case get a constant coefficient linear problem to solve in every sweep.

Using the terminology of Definition 2.1, we now define the well-known Jacobi and Gauss-Seidel splittings.

Definition 2.3 (Jacobi splitting) *A dynamic iteration scheme (11)-(12) for the system (9)-(10) is said to be a Jacobi splitting of the system if* $(\partial \tilde{f}/\partial v) \mid_{\substack{v=u \\ w=u}}$ *equals the diagonal matrix consisting of the diagonal of the matrix* $(\partial f/\partial u) \mid_u$*; whenever the latter exists.*

Definition 2.4 (Gauss-Seidel splitting) *A dynamic iteration scheme (11)-(12) for the system (9)-(10) is said to be a Gauss-Seidel splitting of the system if* $(\partial \tilde{f}/\partial v) \mid_{\substack{v=u \\ w=u}}$ *equals the lower triangular matrix consisting of the lower triangular part of the matrix* $(\partial f/\partial u) \mid_u$*; whenever the latter exists.*

The *Block Jacobi* and *Block Gauss-Seidel* splittings are defined similarly, if we allow for block diagonal and block lower triangular matrices in the definitions above.

The reason why these particular splittings are often used is, besides of their inherent potential for concurrency, that they represent the division of a system into disjoint subsystems. This may be a natural approach in several applications, e.g. when simulating electrical circuits.

For the linear case, when $f(u) = Au$, $A \in \mathbb{R}^{m \times m}$, the Jacobian of f is just A. Even if Definition 2.1 do not rule out the possibility of making $\tilde{f}$ nonlinear, by adding and subtracting nonlinear terms before the splitting, it is in this case most natural to consider dynamic iteration schemes in which $\tilde{f}(v,w) = Mv + Nw$, $M, N \in \mathbb{R}^{m\times m}$; i.e. to use the splitting $A = M + N$.

Note that, even if the above discussion has been dealing with autonomous systems only, it applies equally well to systems in which the right hand side depends explicitly on time.

2.2 Convergence results

We will now derive some convergence results for dynamic iteration for ODE initial value problems. After presenting a convergence theorem for the general case, we look at the convergence of dynamic iteration for linear problems, using splittings $A = M + N$ as described above. Then we look at the situation in which we discretize the dynamic iteration scheme with Runge-Kutta methods.

THEOREM 2.5 (GENERAL CONVERGENCE THEOREM) *Consider an ODE initial value problem (9)-(10), and assume that we have a splitting characterized by the function $\tilde{f}(v,w)$; where $\tilde{f}$ is Lipschitz continuous with respect to both its arguments. Then the dynamic iteration scheme (11)-(12) converges uniformly and ultimately superlinearly, i.e. faster than linear, on all finite intervals $[0,T]$ on which the initial iterate $u^{(0)}$ is bounded.*

Proof. Since $\tilde{f}$ is Lipschitz continuous with respect to both its arguments, it follows immediately from (13) that f is Lipschitz continuous. Therefore (9)-(10) has a unique solution u. Let $(\cdot,\cdot)$ be an inner product on $\mathbb{R}^m$ and $\|\cdot\|$ be the corresponding norm. Let $\tilde{m} \in \mathbb{R}$ be the one-sided Lipschitz constant of $\tilde{f}$ with respect to its first argument, and let $\tilde{N} \geq 0$ be its usual Lipschitz constant with respect to its second argument. That is, we have the following relations

$$\left(\tilde{f}(v_1,w) - \tilde{f}(v_2,w), v_1 - v_2\right) \leq \tilde{m}\|v_1 - v_2\|^2 \ , \ \forall\, v_1, v_2, w \in \mathbb{R}^m \ , \tag{24}$$

$$\|\tilde{f}(v,w_1) - \tilde{f}(v,w_2)\| \leq \tilde{N}\|w_1 - w_2\| \ , \ \forall\, v, w_1, w_2 \in \mathbb{R}^m \ . \tag{25}$$

If $\tilde{m}$ is not known, we can get a (weaker) inequality by using $\tilde{m} = \tilde{M}$ in (24), where $\tilde{M}$ is the usual Lipschitz constant of $\tilde{f}$ with respect to its first argument. If we let $e_k(t) = u^{(k)}(t) - u(t)$ we now get

$$\begin{aligned}
(\dot{e}_{k+1}(t), e_{k+1}(t)) &= \left(\tilde{f}(u^{(k+1)}, u^{(k)}) - \tilde{f}(u,u), u^{(k+1)} - u\right) && (26)\\
&= \left(\tilde{f}(u^{(k+1)}, u^{(k)}) - \tilde{f}(u,u^{(k)}), u^{(k+1)} - u\right) && (27)\\
&\quad + \left(\tilde{f}(u, u^{(k)}) - \tilde{f}(u,u), u^{(k+1)} - u\right) && \\
&\leq \tilde{m}\|e_{k+1}(t)\|^2 + \tilde{N}\|e_k(t)\|\|e_{k+1}(t)\| \ . && (28)
\end{aligned}$$

Assume $\|e_{k+1}(t)\| \neq 0$. Then we have that

$$(\dot{e}_{k+1}(t), e_{k+1}(t)) = \frac{1}{2}\frac{d}{dt}\|e_{k+1}(t)\|^2 = \|e_{k+1}(t)\| \cdot \frac{d}{dt}\|e_{k+1}(t)\| \ ; \tag{29}$$

i.e. for $\|e_{k+1}(t)\| \neq 0$ we have the differential inequality

$$\frac{d}{dt}\|e_{k+1}(t)\| \leq \tilde{m}\|e_{k+1}(t)\| + \tilde{N}\|e_k(t)\| \ . \tag{30}$$

We note that the assumptions of Lipschitz continuity imply that $e_k(t)$ is in C^1, so $\|e_k(t)\|$ is continuous and piecewise in C^1, with possible discontinuities in the derivative for certain times $t_i > 0$. Now we could use the Gronwall lemma (as given in e.g. [4]) in a straightforward manner to get a bound in the form

$$\sup_{t\in[0,T]} \|u^{(k)}(t) - u(t)\| \leq \frac{(\tilde{N}T)^k}{k!} e^{\tilde{m}'T} \cdot \sup_{t\in[0,T]} \|u^{(0)}(t) - u(t)\| \;, \tag{31}$$

where $\tilde{m}' = \max\{0, \tilde{m}\}$; and this would complete the proof. However, a more useful bound is obtainable [24], namely ($\tilde{m} \neq 0$)

$$\sup_{t\in[0,T]} \|u^{(k)}(t) - u(t)\| \leq \left(\frac{\tilde{N}}{-\tilde{m}}\right)^k \Gamma_k(\tilde{m}) \cdot \sup_{t\in[0,T]} \|u^{(0)}(t) - u(t)\| \;, \tag{32}$$

where

$$\Gamma_k(\tilde{m}) = e^{\tilde{m}T} \sum_{j=k}^{\infty} \frac{(-\tilde{m}T)^j}{j!} \;. \tag{33}$$

The superlinear convergence is also predicted by this bound, by noting that

$$\mathcal{O}\left(\frac{\Gamma_k(\tilde{m})}{\Gamma_{k-1}(\tilde{m})}\right) = \mathcal{O}\left(\frac{-\tilde{m}T}{k}\right) \;; \tag{34}$$

i.e. the error reduction from one iteration to the next can not be described by a constant different from zero.

We now prove (32). Let $\omega_k(t) = \|e_k(t)\|$. We argue by induction that

$$\omega_k(t) \leq \left(\frac{\tilde{N}}{-\tilde{m}}\right)^k e^{\tilde{m}t} \sum_{j=k}^{\infty} \frac{(-\tilde{m}t)^j}{j!} \, |\omega_0|_{[0,t]} \;, \tag{35}$$

where $|g(s)|_{[0,t]} = \sup_{s\in[0,t]} |g(s)|$. In fact, assuming that (35) is true for k, we have from (30) that

$$\begin{aligned}
\omega_{k+1}(t) \;&\leq\; \tilde{N} e^{\tilde{m}t} \int_0^t e^{-\tilde{m}s} \omega_k(s)\, ds && (36)\\
&\leq\; \tilde{N} e^{\tilde{m}t} \int_0^t e^{-\tilde{m}s} \left(\frac{\tilde{N}}{-\tilde{m}}\right)^k e^{\tilde{m}s} \sum_{j=k}^{\infty} \frac{(-\tilde{m}s)^j}{j!} \, |\omega_0|_{[0,s]}\, ds && (37)\\
&\leq\; \tilde{N} \left(\frac{\tilde{N}}{-\tilde{m}}\right)^k e^{\tilde{m}t} \sum_{j=k}^{\infty} \frac{(-\tilde{m})^j t^{j+1}}{(j+1)!} \, |\omega_0|_{[0,t]} && (38)\\
&=\; \left(\frac{\tilde{N}}{-\tilde{m}}\right)^{k+1} e^{\tilde{m}t} \sum_{j=k+1}^{\infty} \frac{(-\tilde{m}t)^j}{j!} \, |\omega_0|_{[0,t]} \;. && (39)
\end{aligned}$$

Since the right hand side of (35) is non-decreasing, (32) follows. □

If one has a priori bounds for $\sup_{t\in[0,T]} \|u^{(0)}(t) - u(t)\|$, equations (31) and (32) can be used as error bounds for the iteration. In (31), the bound can grow large for small k, and does not really take into account in what way the system is decoupled. The bound (32) is better, since it shows clearly how the splitting, characterized by $\tilde{m}$ and $\tilde{N}$, affects the convergence. To be more precise, we see that if $\tilde{m} < 0$, i.e. we solve a dissipative system in each iteration, we have

$$0 < \Gamma_{k+1}(\tilde{m}) < \Gamma_k(\tilde{m}) < 1 \; ; \tag{40}$$

which, if in addition $-\tilde{m} > \tilde{N}$, implies that the iteration is a contraction in the sup-norm on all finite intervals $[0, T]$. If $\tilde{m} > 0$, the sign of $\Gamma_k(\tilde{m})$ is $(-1)^k$, and we have

$$|\Gamma_k(\tilde{m})| < e^{\tilde{m}T} \; ; \tag{41}$$

which, if in addition $\tilde{m} > \tilde{N}$, assures that the initial error will never grow more than by a factor $e^{\tilde{m}T}$. In any case, (32) indicates that we should choose $\tilde{f}$ such that the ratio $\tilde{N}/|\tilde{m}|$ is as small as possible, i.e. that we minimize the strength of the coupling between consecutive iterates. If the ratio is greater than 1, we should expect growth in the errors in the first iterations, before the superlinear convergence is noticed.

For the linear case where $f(u) = Au$, $A \in \mathbb{R}^{m\times m}$, we can write the bound (32) as

$$\sup_{t\in[0,T]} \|u^{(k)}(t) - u(t)\| \leq \left(\frac{\|N\|}{-\mu[M]}\right)^k \Gamma_k(\mu[M]) \cdot \sup_{t\in[0,T]} \|u^{(0)}(t) - u(t)\| \; , \tag{42}$$

where $\mu[M]$ is the logarithmic norm of the matrix M; i.e. the best possible choice for $\tilde{m}$. In the linear case, explicit solutions can be found, and the iteration can be written

$$u^{(k+1)} = \mathcal{K}u^{(k)} + \varphi \; , \tag{43}$$

where $\mathcal{K}$ is a linear integral operator. Due to the superlinear convergence, the spectral radius of this operator is zero on all finite intervals. However, analyzing the iteration in L^p-spaces on $[0, \infty)$, Miekkala and Nevanlinna [12] get the following result (adapted to our notation):

Theorem 2.6 (Spectral radius of $\mathcal{K}$) *Let $X = L^p([0,\infty); \mathbb{R}^m)$ with $1 \leq p \leq \infty$, and assume that all eigenvalues of M have negative real parts. Then*

$$r_\sigma(\mathcal{K}) = \max_{\xi\in\mathbb{R}} \rho[(i\xi I - M)^{-1}N] \, . \tag{44}$$

In the theorem, ρ denotes the spectral radius of a matrix, and $r_\sigma(\mathcal{K}) = \lim_{k\to\infty} \|\mathcal{K}^k\|_p^{1/k}$; i.e. the spectral radius of $\mathcal{K}$ in X. As we can see, the choice of L^p space do not affect $r_\sigma(\mathcal{K})$. In this way we get a non-zero expression for the asymptotic average linear convergence rate $r_\sigma(\mathcal{K})$, and it depends only on the splitting matrices M and N. Hence, we can use this quantity in order to compare the convergence rates of different splittings. Miekkala and Nevanlinna compute this quantity for several types of splittings.

Skålin [22] analyzes the case in which we split the linear system like above and discretize it by Runge-Kutta methods. The main result is the following.

Theorem 2.7 (Convergence of discretized system) *For all M, N such that $A = M + N$ and M possesses only real eigenvalues, there exists an $h^* > 0$ such that the dynamic iteration scheme discretized by any Runge-Kutta method converges.*

In addition, Skålin gives some convergence results in the spirit of the continuous cases given in [12].

As mentioned above, when e.g. $\tilde{N}/|\tilde{m}| > 1$ the errors may grow large before the superlinear convergence starts, and this can cause numerical difficulties in the computations. In practical situations, one may therefore want to split the interval of integration $[0, T]$ into subintervals, or *windows*, $[0, T_1] \cup [T_1, T_2] \cup \cdots \cup [T_{n-1}, T_n]$; and use the dynamic iteration procedure on each interval in turn, in order to control the error growth within each interval. Obviously, we need estimates for the length of such windows, and such estimates can be found in several ways. One possibility is to use an adaptive approach, by iterating on shorter and shorter intervals until we end up with an interval giving a satisfactory error decay. Another possibility is to make use of the error bound (31), and solve for the T that will give the wanted error decay function. We see that by making T small enough, contractivity is assured. In the linear case, we have an estimate that makes use of the *convergence abscissa.* The convergence abscissa β is defined as the infimum of all α's for which $r_\sigma(\mathcal{K}) < 1$ in the space $C([0,\infty);\mathbb{R}^m)$ normed by $\|u\|_{\alpha,\infty} = \|e^{-\alpha t}u(t)\|_\infty$, $\alpha > 0$; i.e. β represents the least amount of exponential damping needed for the iteration to converge uniformly on $[0, \infty)$. Now we can follow the analysis of Leimkuhler [8]:

$$
\begin{aligned}
\|\mathcal{K}^k u\|_T &= \|e^{\beta t} e^{-\beta t} \mathcal{K}^k u\|_T && (45)\\
&\leq e^{\beta T} \sup_{\tau \in [0,T]} \|e^{-\beta\tau}\mathcal{K}^k u\| && (46)\\
&\leq e^{\beta T} \sup_{\tau \in [0,\infty)} \|e^{-\beta\tau}\mathcal{K}^k u\| && (47)\\
&= e^{\beta T} \|\mathcal{K}^k u\|_{\beta,\infty} && (48)\\
&\approx e^{\beta T} [r_\sigma(\mathcal{K})]^k \|u\|_{\beta,\infty} && (49)\\
&= e^{\beta T} \|u\|_{\beta,\infty}\ ; && (50)
\end{aligned}
$$

since the supremum in the definition of β will always be equal to 1, if it exists. We note that any Lipschitz continuous function u on $[0, T]$ can be extended Lipschitz continuously into $[0, \infty)$, such that $\|u\|_{\beta,\infty} \leq \|u\|_T$. Hence, if the approximation of $\|\mathcal{K}^k u\|_{\beta,\infty}$ by $[r_\sigma(\mathcal{K})]^k \|u\|_{\beta,\infty}$ is reasonable, we can choose the window length as e.g. $1/\beta$; and this will then assure only moderate growth in the iterations on $[0, 1/\beta]$.

Lastly, we note that it is also possible to define dynamic iteration schemes for second order ODE initial value problems. Dynamic iteration schemes especially designed for second order equations are often more natural to use, compared to first order schemes applied to the first order, twice as large, equivalent system. Convergence results for the linear case, for both continuous and discretized dynamic iteration schemes, can be found in [2].

3 Generalizing to time-dependent PDEs

We will now look at the possibility for using dynamic iteration techniques, or rather generalizations of the dynamic iteration idea, in the process of solving time-dependent partial differential equations. We divide the discretization process into two parts, as shown in Figure 2. The first step consists of spatial discretization of the PDE, i.e. the approximation of the spatial derivatives by e.g. spectral collocation or finite differences. After this step, we are

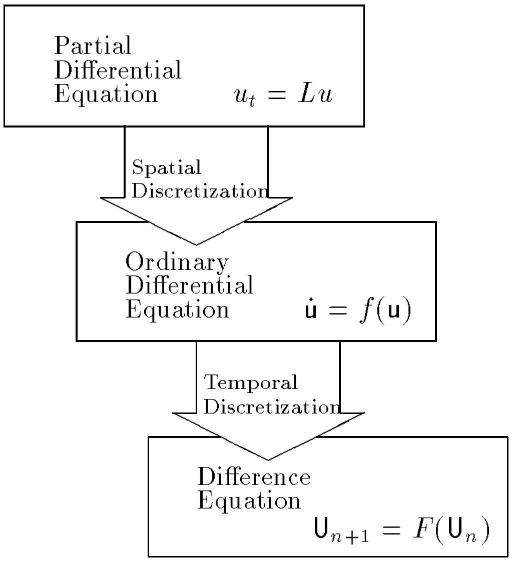

Figure 2: The discretization process for time-dependent PDEs

left with an ODE system in time. The next step consists of time discretization of the ODE, which in practice means applying e.g. a Runge-Kutta method. Since the ODE system is likely to get stiff when the size increases, implicit integration methods are interesting alternatives here.

The possibility for application of usual dynamic iteration techniques, as described in the last section, to these ODE systems is evident. However, by generalizing the idea of introducing an artificial decoupling, or splitting, to the PDE case, we can imagine another kind of dynamic iteration technique, now for PDEs. In this case, we could apply dynamic iteration techniques both before and after spatial discretization of the PDE, and together these two techniques could introduce a substantial amount of concurrency in the numerical solution process. In addition, we would expect each independent subsystem to be integrated more efficiently than if they were parts of a larger system, as is the case for dynamic iteration for ODEs.

3.1 Outer Dynamic Iteration (ODI)

When applying dynamic iteration to an ODE system, what we do is to introduce an artificial decoupling, or splitting, to the system; transforming the original problem into several smaller problems. In order to retain the information lost in the decoupling, we have to perform several sweeps for the smaller problems. In the PDE context, a natural artificial decoupling comes from a domain decomposition approach. In [18], Quarteroni discusses iterative schemes based on domain decomposition for time-discretized systems; i.e. when the iteration takes place within the scope of a single time step. The coupling between adjacent subdomains is read from the previous iteration, and when convergence is achieved the same approach is applied to the next time level. It is possible, however, to apply this kind of splitting approach before any discretizations are made; i.e. to apply it to the continuous PDE system. The idea consists of decoupling the equations through domain decomposition over the time interval of interest, reading the values that couple adjacent subdomains from the previous sweep. The similarity to the ideas of dynamic iteration methods for ODEs, makes it reasonable to consider this as a dynamic iteration method; and we will in the following refer to this as an *Outer Dynamic Iteration technique* (ODI).

For hyperbolic problems, we can imagine both disjoint and non-disjoint domain decompositions. In any case, we should make use of the information given by the characteristics in order to determine when to read incoming information from the neighbouring domain from the previous sweep. There should be no reflection of outgoing information on the interfaces to the other subdomains, as these boundaries are artificial. In section 4, we will present a specific ODI algorithm for hyperbolic PDEs, but in general the technique is developed as follows. Consider the following PDE initial-boundary problem,

$$
\begin{aligned}
(\frac{\partial}{\partial t} + L)u = f \quad &, \quad (x,t) \in \Omega \times [0,T] && (51)\\
Bu = b \quad &, \quad (x,t) \in \partial\Omega \times [0,T] && (52)\\
u = g \quad &, \quad (x,t) \in \Omega \times \{0\} \,. && (53)
\end{aligned}
$$

Schematically, we now formulate the problem on subdomains Ω_i of the original domain, where $\cup_i \overline{\Omega}_i = \overline{\Omega}$. If we let the set of neighbours $\mathcal{N}_i$ of Ω_i be defined by

$$
j \in \mathcal{N}_i \ \text{ iff } \ \overline{\Omega}_j \cap \overline{\Omega}_i \neq \emptyset \,,\ j \neq i \,, \qquad (54)
$$

we get

$$
\begin{aligned}
(\frac{\partial}{\partial t} + L)u_i = f \quad &, \quad (x,t) \in \Omega_i \times [0,T] && (55)\\
\mathcal{M}(u_i, u_j) = 0 \quad &, \quad (x,t) \in \partial\Omega_i \cap \overline{\Omega}_j \times [0,T] \,,\ \forall\, j \in \mathcal{N}_i && (56)\\
Bu_i = b \quad &, \quad (x,t) \in \partial\Omega_i \cap \partial\Omega \times [0,T] && (57)\\
u_i = g \quad &, \quad (x,t) \in \Omega_i \times \{0\} \,; && (58)
\end{aligned}
$$

where $\mathcal{M}$ is an interface operator coupling the solutions on the different subdomains. The ODI method then appears by decoupling the problems, i.e. suppressing the couplings through the interface operator:

$$
\begin{aligned}
(\frac{\partial}{\partial t} + L)u_i^{(k+1)} = f \quad &, \quad (x,t) \in \Omega_i \times [0,T] && (59)\\
\mathcal{M}(u_i^{(k+1)}, u_j^{(k)}) = 0 \quad &, \quad (x,t) \in \partial\Omega_i \cap \overline{\Omega}_j \times [0,T] \,,\ \forall\, j \in \mathcal{N}_i && (60)\\
Bu_i^{(k+1)} = b \quad &, \quad (x,t) \in \partial\Omega_i \cap \partial\Omega \times [0,T] && (61)\\
u_i^{(k+1)} = g \quad &, \quad (x,t) \in \Omega_i \times \{0\} \,; && (62)
\end{aligned}
$$

where $k = 0, 1, 2, \ldots$. The initial iterates $u_i^{(0)}$ can be chosen as e.g. the initial value $g\,|_{\Omega_i}$. By decoupling this way, we split the system (51)-(53) into several smaller independent PDEs, and these can therefore be solved in parallel on $[0,T]$ in each sweep; i.e. for each k. Figure 3 shows the application of the ODI technique schematically.

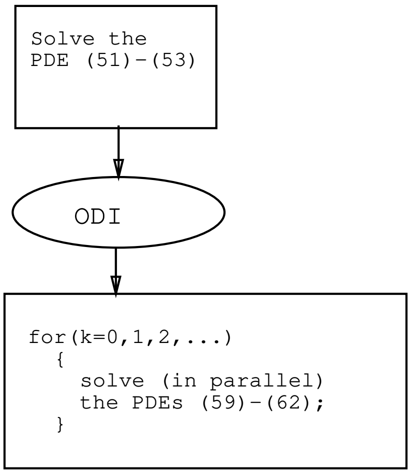

Figure 3: The ODI technique

3.2 Inner Dynamic Iteration (IDI)

Assume now that we have completed the spatial discretization of our PDE, and that we are left with an ODE initial value problem (where we denote the discrete version of u by u)

$$\dot{\mathsf{u}} = f(\mathsf{u}) \,, \tag{63}$$

$$\mathsf{u}(0) = \mathsf{u}_0 \,. \tag{64}$$

We will discuss in Section 6 how to obtain these ODEs for the hyperbolic case. Obviously, the dynamic iteration techniques from Definition 2.1 can be applied to these special ODEs, appearing in the discretization process of time-dependent PDEs, and we will in this context refer to the techniques as *Inner Dynamic Iteration techniques* (IDI). The choice of the names ODI and IDI are motivated by the fact that the ODI technique affects only the couplings to other domains, or the outside world of each domain; while the IDI technique affects only the couplings within the domain.

We could also define two-periodic IDI schemes for the ODEs. As an example, consider the linear equation

$$u_t = Au_x + Bu_y \,, \tag{65}$$

which we discretize spatially by spectral collocation, yielding (not taking into consideration the effect of the boundary values)

$$\dot{\mathsf{u}} = A \otimes (I \otimes D)\mathsf{u} + B \otimes (D \otimes I)\mathsf{u} \,; \tag{66}$$

where D is the spectral differentiation matrix and I is the identity matrix of dimension equal to the total number of points in the grid. Now, define the following dynamic iteration scheme:

$$\begin{aligned} \dot{\mathsf{u}}^{(k+1/2)} &= A \otimes (I \otimes D)\mathsf{u}^{(k+1/2)} + B \otimes (D \otimes I)\mathsf{u}^{(k)} \,, && (67) \\ \dot{\mathsf{u}}^{(k+1)} &= A \otimes (I \otimes D)\mathsf{u}^{(k+1/2)} + B \otimes (D \otimes I)\mathsf{u}^{(k+1)} \,, && (68) \end{aligned}$$

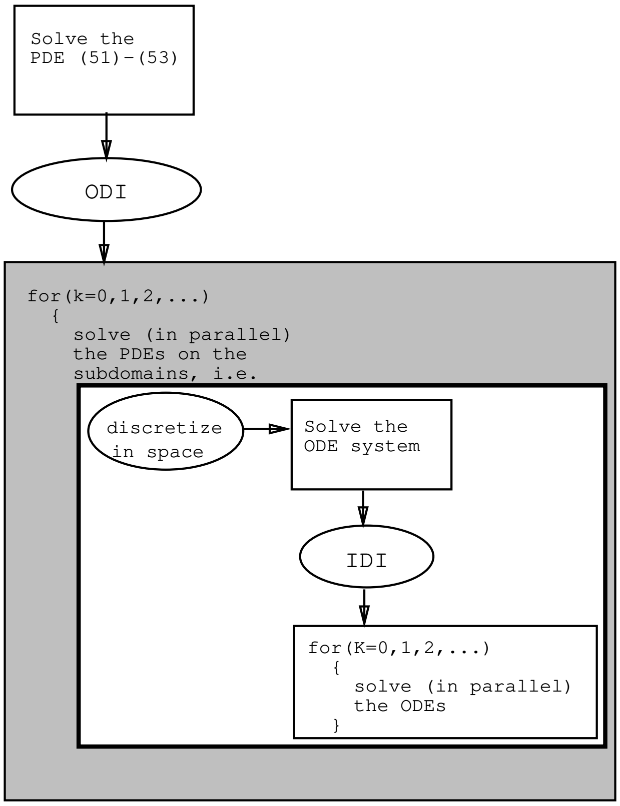

Figure 4: The composition of ODI and IDI techniques

$$\mathsf{u}^{(k+1/2)}(0) = \mathsf{u}^{(k+1)}(0) = \mathsf{u}_0 \ ; \tag{69}$$

for $k = 0, 1, 2, \ldots$. This would define an alternating direction IDI method, in which we suppress the couplings to the x-derivatives and the y-derivatives alternately.

Summing up, we have defined two classes of dynamic iteration methods in the process of solving time-dependent PDEs. The ODI techniques are domain decomposition type methods to be applied before spatial discretization, and the IDI techniques apply to the ODEs appearing after spatial discretization. Both methods can introduce splittings to the system, and therefore enable parallel processing. One could imagine two such methods to be combined, giving two levels of splittings; both between the subdomains and within each subdomain. This would result in a highly parallel iterative scheme, and this is shown schematically in Figure 4.

4 An ODI algorithm for hyperbolic PDEs

Let us now formulate an ODI algorithm for quasilinear hyperbolic PDEs. Following the lines of the generic development of section 3.1, we write our initial-boundary problem as

$$(\frac{\partial}{\partial t} + L)u = f \quad , \quad (x,t) \in \Omega \times [0,T] \tag{70}$$

$$Bu = b \quad , \quad (x,t) \in \partial\Omega \times [0,T] \tag{71}$$

$$u = g \quad , \quad (x,t) \in \Omega \times \{0\} \; ; \tag{72}$$

where $L : u \mapsto \sum_{j=1}^{d} A_j(u,x,t)u_{x_j} + C(u,x,t)u$ is such that $(\frac{\partial}{\partial t} + L)$ is a first order hyperbolic differential operator, $\Omega \subset \mathbb{R}^d$ is open, and

$$u : \overline{\Omega} \times [0,T] \to \mathbb{R}^m \; , \tag{73}$$

$$A_j, C : \mathbb{R}^m \times \overline{\Omega} \times [0,T] \to \mathbb{R}^{m \times m} \; . \tag{74}$$

The matrix valued functions A_j and C are supposed to be continuous. The boundary condition (71) must be chosen such that the problem is well posed. In order to specify this more closely, let us begin by defining the *characteristic matrix in direction* ξ as

$$A(\xi) = \sum_{j=1}^{d} A_j(u,x,t)\xi_j \; ; \tag{75}$$

where $\xi \in \mathbb{R}^d$, $\|\xi\| = 1$. Since the system is hyperbolic, $A(\xi)$ is diagonalizable for all ξ. In other words, if $S(\xi)$ is the matrix with the left linearly independent eigenvectors of $A(\xi)$ as rows, and $\Lambda(\xi)$ is the diagonal matrix containing the corresponding eigenvalues λ_i, we have

$$S(\xi)A(\xi) = \Lambda(\xi)S(\xi) \; . \tag{76}$$

Now, choose $\{\tau^{(k)}\}_{k=1}^{d-1}$, $\tau^{(k)} \in \mathbb{R}^d$, such that $\{\xi, \tau^{(1)}, \ldots, \tau^{(d-1)}\}$ constitute an orthonormal basis for $\mathbb{R}^d$. We then make a change of coordinates in (70), and get the equivalent system

$$S(\xi)u_t + \Lambda(\xi)S(\xi)u_\xi = S(\xi)\left[f - \sum_{j=1}^{d} A_j(u,x,t)\sum_{k=1}^{d-1} u_{\tau^{(k)}}\tau_j^{(k)} - C(u,x,t)u\right] \; . \tag{77}$$

These equations are called the *compatibility conditions* for the system, with respect to direction ξ (see e.g. [17]). We can now classify the equations with respect to whether they represent flow of information in direction $+\xi$ or $-\xi$. We notice that the equations in system (77) describe flow of information along bicharacteristics in the characteristic manifolds of dimension d through the point (p,t) in $\Omega \times [0,T]$ where we are situated (cf. [6]). The projections of the normal vectors of these hypersurfaces in Ω are parallel to ξ in p, and the normal vectors themselves are given by $c_i(-\lambda_i, \xi) \in \mathbb{R}^{d+1}$; where $c_i > 0$ are normalization factors. So if λ_i is positive, this normal vector points in direction of decreasing t, meaning that equation i in (77) describes flow of information in the same direction as ξ, Conversely, if λ_i is negative, the corresponding equation describes information flow in the opposite direction of ξ. See Figure 5 for an illustration. If ξ_p is the outwards normal in a point p on $\partial\Omega$, we should, in order to achieve a well posed problem, at every instant specify m_2 boundary conditions, where m_2 is the number of negative eigenvalues of $A(\xi_p)$. In other words, we must require that B has rank

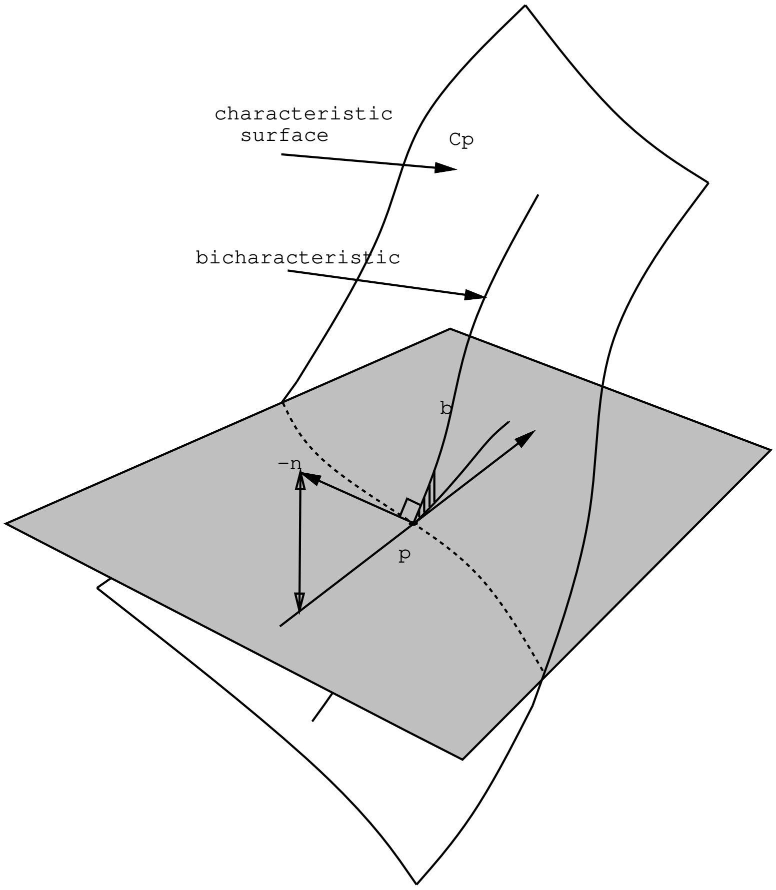

Figure 5: The concept of characteristic surfaces and bicharacteristics.

m_2. It is often natural to specify the incoming *characteristic quantities*, given by $S_- u$; where S_- is the submatrix of $S(\xi_p)$ corresponding to the negative eigenvalues of $A(\xi_p)$. That is, we partition the matrix $\Lambda(\xi)$ as

$$\Lambda(\xi) = \begin{bmatrix} \Lambda_+ & 0 \\ 0 & \Lambda_- \end{bmatrix} , \tag{78}$$

where $\Lambda_+ \in \mathbb{R}^{m_1 \times m_1}$ contains the m_1 non-negative eigenvalues of $A(\xi)$, and $\Lambda_- \in \mathbb{R}^{m_2 \times m_2}$ contains the m_2 negative eigenvalues, and let S be partitioned accordingly as

$$S(\xi) = \begin{bmatrix} S_+ \\ S_- \end{bmatrix} . \tag{79}$$

We now split the domain into M regular disjoint subdomains $\{\Omega_j\}_{j=1}^{M}$, where $\Omega = \overset{\circ}{\overline{\cup_i \Omega_i}}$, and let the neighbours of domain i be described by the set $\mathcal{N}_i$. In a point p on the boundary between two subdomains, say Ω_i and Ω_j, we make use of the compatibility conditions (77) in order to specify boundary conditions for the two subdomains. That is, if ξ_p is the normal vector of $\partial\Omega_i$, pointing into Ω_j, we specify the characteristic quantities $S_- u$ coming into Ω_i. The natural place to look for input values is in the adjacent domain Ω_j. Conversely, we specify for Ω_j the incoming characteristic quantities by reading values from Ω_i. Note that in problems where the coefficient matrices A_j depend on u, we do not in general have that $A(\xi_p)$ computed from Ω_i, equals $-A(-\xi_p)$ computed from Ω_j; so we have no guarantee a priori that the 'outgoing information' from Ω_i enters Ω_i.

Hence, our formulation of the original problem (70)-(72) as problems on the subsets Ω_i become

$$\left(\frac{\partial}{\partial t} + L\right) u_i = f \quad , \quad (x,t) \in \Omega_i \times [0,T] \tag{80}$$

$$S_- u_i = S_- u_j \quad , \quad (x,t) \in \partial\Omega_i \cap \overline{\Omega}_j \times [0,T] \, , \; \forall \, j \in \mathcal{N}_i \tag{81}$$

$$B u_i = b \quad , \quad (x,t) \in \partial\Omega_i \cap \partial\Omega \times [0,T] \tag{82}$$

$$u_i = g \quad , \quad (x,t) \in \Omega_i \times \{0\} \, ; \tag{83}$$

where S_- is the matrix valued function of u_i, x, t and ξ, as defined above.

To justify the domain decomposition we have some lemmas. Suppose that the original problem (70)-(72) on Ω is well posed, i.e. that it has a unique solution for arbitrary data, depending continuously on the data. Let $L = L(x,t)$ and $B = B(x,t)$ be linear, f and b be functions of x and t only; and let Ω be decomposed into M disjoint subdomains Ω_i as above. Let $\Gamma = \cup_i \partial\Omega_i \backslash \partial\Omega$, and $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$. Assume Γ to be piecewise smooth, in the meaning that the the set $\Gamma_\xi = \{p \in \Gamma \mid \xi_p \text{ exists } \} \subset \Gamma$ is dense in Γ, where ξ_p is the normal vector of Γ at the point p. In addition, for each t, we assume that the set $\Gamma_A = \{p \in \Gamma_\xi \mid \det(A(\xi_p)) \neq 0\} \subset \Gamma$ is dense in Γ. Then we can state the following lemma.

LEMMA 4.1 *If $\{u_i\}$ is a solution to the decomposed problem (80)-(83), where each u_i is of class $C^1([0,T]; C^1(\overline{\Omega}_i))$, then $\{u_i\}$ is the only such solution.*

Proof. Assume that we have two solutions $\{u_i\}$ and $\{v_i\}$ of the decomposed problem. We see that $\{w_i\} = \{u_i - v_i\}$ satisfies the following system

$$\left(\frac{\partial}{\partial t} + L\right) w_i = 0 \quad , \quad (x,t) \in \Omega_i \times [0,T] \tag{84}$$

$$S_-w_i = S_-w_j \quad , \quad (x,t) \in \Gamma_{ij} \times [0,T] \; , \; \forall\, j \in \mathcal{N}_i \tag{85}$$
$$Bw_i = 0 \quad , \quad (x,t) \in \partial\Omega_i \cap \partial\Omega \times [0,T] \tag{86}$$
$$w_i = 0 \quad , \quad (x,t) \in \Omega_i \times \{0\} \; . \tag{87}$$

For $p \in \Gamma_{ij} \cap \Gamma_\xi$, and ξ_p is the normal vector pointing from Ω_i to Ω_j, the interface conditions (85) for each domain yield

$$S_-(p,t,\xi_p)w_i = S_-(p,t,\xi_p)w_j \;\; , \;\; S_-(p,t,-\xi_p)w_j = S_-(p,t,-\xi_p)w_i \; . \tag{88}$$

Since $S_-(p,t,-\xi_p) = S_+(p,t,\xi_p)$, and $S(p,t,\xi_p)$ is non-singular by the hyperbolicity of the problem, we get

$$S(p,t,\xi_p)w_i = \begin{bmatrix} S_+(p,t,\xi_p) \\ S_-(p,t,\xi_p) \end{bmatrix} w_i = \begin{bmatrix} S_+(p,t,\xi_p) \\ S_-(p,t,\xi_p) \end{bmatrix} w_j = S(p,t,\xi_p)w_j \; ; \tag{89}$$

which implies $w_i = w_j$ in p, for all $t \in [0,T]$. For $p \in \Gamma_{ij}$, $p \notin \Gamma_\xi$, assume that $w_i \neq w_j$. But since w_i and w_j are continuous on Γ_{ij}, there must be a neighbourhood of p in Γ_{ij} where $w_i \neq w_j$, leading to a contradiction since Γ_ξ is dense in Γ. Hence, $w_i = w_j$ in $\Gamma_{ij} \times [0,T]$, and thus w, defined by its restrictions $w\,|_{\overline{\Omega}_i} = w_i$, is well defined and continuous on $\overline{\Omega} \times [0,T]$.

Now it is clear that w is of class $C^1([0,T];C(\overline{\Omega}))$. It satisfies the differential equation $(\partial/\partial t + L)w = 0$ in $\Omega_i \times [0,T]$, and it satisfies the boundary condition $Bw = 0$ on $\partial\Omega \times [0,T]$ and the initial condition $w = 0$ in Ω. We will now show that it is in fact of class $C^1([0,T];C^1(\overline{\Omega}))$, and satisfies the differential equation also on $\Gamma \times [0,T]$. Let $p \in \Gamma_{ij} \cap \Gamma_A$, and let ξ_p be a normal vector for Γ at p. Now, make a local change of coordinates from $(x_1,\ldots,x_d)$ to $(\xi,\tau^{(1)},\ldots,\tau^{(d-1)})$ such that $\xi_p = (1,0,\ldots,0)$ in the new coordinates. Since $w_i = w_j$ on $\Gamma_{ij} \times [0,T]$ and $w_i \in C^1([0,T];C^1(\overline{\Omega}_i))$, $w_j \in C^1([0,T];C^1(\overline{\Omega}_j))$, the first partial derivatives of w with respect to $t,\tau^{(1)},\ldots,\tau^{(d-1)}$ are continuous across Γ_{ij}. Now, note that by multiplying the compatibility condition (77) by S^{-1} we get, in Ω_i and Ω_j,

$$A(x,t,\xi_p)(w_i)_\xi = Q(x,t,w,w_{\tau^{(k)}}) \;\; , \;\; A(x,t,\xi_p)(w_j)_\xi = Q(x,t,w,w_{\tau^{(k)}}) \tag{90}$$

respectively, where Q is linear and x is in a neighbourhood of p in Ω where $A(x,t,\xi_p)$ is non-singular. Such a neighbourhood must exist, since $\det(A(x,t,\xi_p))$ is a continuous function of x. Letting x approach p from both Ω_i and Ω_j in this expression, we get that the left hand sides approaches the same limit from both sides, since all the quantities in the right hand sides are continuous across Γ. That is, $A(p,t,\xi_p)(w_i)_\xi = A(p,t,\xi_p)(w_j)_\xi$, implying $(w_i)_\xi = (w_j)_\xi$ in p. By the density of Γ_A in Γ, we have

$$(w_i)_\xi = (w_j)_\xi \text{ on } \Gamma_{ij} \; . \tag{91}$$

Hence, all the first partial derivatives of w exists and are continuous not only in $\{\Omega_i\}$, but in the entire domain Ω. In addition, since $A(p,t,\xi_p)w_\xi = Q(p,t,w,w_{\tau^{(k)}})$ for all $p \in \Gamma$, we have that the differential equation is satisfied on Γ, and thus in the entire domain Ω.

We now conclude that w is of class $C^1([0,T];C^1(\overline{\Omega}))$, and that it solves the system

$$(\frac{\partial}{\partial t} + L)w = 0 \quad , \quad (x,t) \in \Omega \times [0,T] \tag{92}$$
$$Bw = 0 \quad , \quad (x,t) \in \partial\Omega \times [0,T] \tag{93}$$
$$w = 0 \quad , \quad (x,t) \in \Omega \times \{0\} \; . \tag{94}$$

But this system is assumed to have a unique solution, and thus we must have $w \equiv 0$ in $\overline{\Omega} \times [0,T]$. This proves our uniqueness result, since then $w_i = u_i - v_i \equiv 0$ for $i = 1, \ldots, M$, giving $u_i \equiv v_i$ in $\overline{\Omega} \times [0,T]$. □

Corollary 4.2 *The function u defined by $u \mid_{\overline{\Omega}_i} = u_i$ is the solution of the original problem, and it is of class $C^1([0,T]; C^1(\overline{\Omega}))$.*

Proof. The reasoning in the proof of the lemma which leads to the conclusion that w is the unique solution of the original problem with zero data, applies directly to this case, where we have possibly non-zero data. □

The existence of classical solutions of (80)-(83) is connected to the existence of a classical solution of the original problem through the following lemma.

Lemma 4.3 *Let u be a solution of a quasilinear system (70)-(72) of class $C^1([0,T]; C^1(\overline{\Omega}))$. Then the $\{u_i\}$, given by $u_i = u \mid_{\overline{\Omega}_i}$, are of class $C^1([0,T]; C^1(\overline{\Omega}_i))$, and they solve the problem (80)-(83).*

Proof. It is clear that u_i is of class $C^1([0,T]; C^1(\overline{\Omega}_i))$, since $\Omega_i \subset \Omega$. Hence it satisfies the differential equation in Ω_i, together with the restriction of the initial condition and the boundary condition to Ω_i. It remains to show that the interface conditions are satisfied. But this follows immediately, since $u_i = u = u_j$ on Γ_{ij} implies $S_-(u_i, p, t, \xi_p)u_i = S_-(u_i, p, t, \xi_p)u = S_-(u_i, p, t, \xi_p)u_j$ for $p \in \Gamma_{ij}$. □

The lemma assumes that the solution u is continuously differentiable in both t and x. We have the following result from [20], concerning linear systems.

Lemma 4.4 *Under certain conditions on the coefficients of a system (70)-(72), we have that if $g \in H^s(\Omega)$, $f \in H^s([0,T] \times \Omega)$ and $b \in H^{s+1/2}([0,T] \times \partial\Omega)$, then the solution u belongs to $C^r([0,T]; H^{s-r}(\Omega))$ for $0 \leq r \leq s$ if and only if*

$$\sum_{i=0}^{p} \left(\begin{array}{c} p \\ i \end{array} \right) (\partial_t^i B)(0) g_{p-i} = \partial_t^p b(0) \ , \ \ (x,t) \in \partial\Omega \times \{0\} \ \textit{for } 0 \leq p \leq s-1. \tag{95}$$

Here, $\partial_t = (\partial/\partial t)$ and

$$g_p = -\sum_{i=0}^{p-1} \left(\begin{array}{c} p-1 \\ i \end{array} \right) L_i(0) g_{p-1-i} + (\partial_t^{p-1} f)(0) \,, \tag{96}$$

where $g_0 = g$ and $L_0 = L$, $L_i = \partial_t L_{i-1} = \sum(\partial_t^i A_j(x,t))\partial_{x_j} + \partial_t^i C(x,t)$.

Condition (95) is also called a compatibility condition, and should not be confused with our earlier notion of compatibility conditions. Assume that (95) is satisfied for some s. Using the fact that $H^m(\Omega) \subset C(\overline{\Omega})$ when $m > d/2$ with continuous injection, whenever Ω is a open subset of $\mathbb{R}^d$ with C^0 boundary (see [21]), we then have that $u \in C^r([0,T]; C^k(\overline{\Omega}))$ whenever $s - r > d/2 + k$; $s - r \in \mathbb{N}$. In particular, taking $r = 1$, this means that $u \in C^1([0,T]; C^1(\overline{\Omega}))$ whenever $s > d/2 + 2$.

The ODI method now appears if we decouple the problems on the subdomains by suppressing the couplings to the neighbouring domains, that is

$$(\frac{\partial}{\partial t} + L)u_i^{(k+1)} = f \quad , \quad (x,t) \in \Omega_i \times [0,T] \tag{97}$$

$$S_- u_i^{(k+1)} = S_- u_j^{(k)} \quad , \quad (x,t) \in \partial\Omega_i \cap \overline{\Omega}_j \times [0,T] \, , \; \forall j \in \mathcal{N}_i \tag{98}$$

$$B u_i^{(k+1)} = b \quad , \quad (x,t) \in \partial\Omega_i \cap \partial\Omega \times [0,T] \tag{99}$$

$$u_i^{(k+1)} = g \quad , \quad (x,t) \in \Omega_i \times \{0\} \; ; \tag{100}$$

where $k = 0, 1, 2, \ldots$. It is now easy to see that we have obtained M independent PDE systems in every iteration (or sweep); one for each subdomain.

REMARK 4.5 Note the analogy to the method proposed by Quarteroni [19]. He proposes a method with the same decoupling, but where the decoupling is only valid within each step in the fully discretized equations; i.e. that one should iterate on time level n until the desired accuracy is achieved, before proceeding to time level $n+1$. Using the language of relaxation methods for ODEs, his method could be called a time-point iteration method; while the one we have proposed above is a continuously defined dynamic iteration method, since the iterates here are functions on $\Omega_i \times [0,T]$.

We can now spatially discretize the equations by e.g. the spectral collocation method. This is done by mapping the subdomains to a d-dimensional cube, and performing all calculations in this regular domain; see e.g. Kopriva [7]. We then make use of the ODE formulation of such equations, introduced below, in order to get properly defined ODEs on each of the subdomains. In other words, we will after spatial discretization get the iteration

$$\dot{\mathsf{u}}_i^{(k+1)} = f(\mathsf{u}_i^{(k+1)}, \mathsf{u}_{\mathcal{N}_i}^{(k)}) \; , \; i = 1, \ldots, M \; , \; k = 0, 1, 2, \ldots \; , \tag{101}$$

with the initial values $\mathsf{u}_i^{(k+1)}(0)$ given by the initial condition (72). We thus have to solve M independent ODE systems in each iteration, which yields an obvious potential for parallelization. We could in addition apply IDI methods to these ODEs, e.g. a Jacobi splitting, which would leave us to solve $M \cdot N$ single independent ODEs in each iteration; where N is the number of collocation points in each subdomain.

We should note that the communication between the different processors is limited to data transfer after each ODI sweep; which would add up to a time consumption that is negligible as opposed to the time required to calculate the sweeps (at least for window sizes of some length). However, large amounts of data have to be stored on each processor during a sweep (even more so in the IDI case!), and this is without any doubt the main drawback of these methods.

Let us now make some investigations on the efficiency of the ODI algorithm compared to the single domain approach. We assume that we solve a system of m equations in the domain $(-1,1)^d \times [0,T]$, and that we use the Chebyshev collocation method with matrix multiplication to approximate the space derivatives. We require ν collocation points in each direction, giving a total of $N = \nu^d$ collocation points. It is reasonable to assume that the cost of a function evaluation in the ODE solver is dominated by the cost $C_F(\nu) = dm\nu^{d+1}$ of the spectral differentiation. Using an explicit Runge-Kutta method with constant step size h, the sequential time consumption of the direct solver would then be

$$T_{exp,dir} = C_e(h,T) \cdot dm\nu^{d+1} \; . \tag{102}$$

If we use the ODI approach with μ subdivisions of Ω in each direction, and a total of $M = \mu^d$ parallel sweeps, the time consumption (neglecting communication) would be

$$T_{exp,ODI} = \mu^d \cdot C_e(h,T) \cdot dm(\frac{\nu}{\mu})^{d+1} = \frac{1}{\mu} \cdot T_{exp,dir} \; ; \tag{103}$$

i.e. we would get a speedup $\mathbb{S}_{par} = \mu$.

Since large systems are likely to get stiff, we might want to use an implicit Runge-Kutta method to solve the problem. Neglecting the cost of factoring the Jacobian, the sequential time consumption of the direct solver would now be

$$T_{imp,dir} = C_i(h,T) \cdot [dm\nu^{d+1} + (m\nu^d)^2] \; ; \tag{104}$$

where $C_{LU}(\nu) = (m\nu^d)^2$ is the cost of solving the linear system by forward elimination and back substitution. Using the ODI method with $M = \mu^d$ parallel sweeps as above, the time consumption becomes

$$T_{imp,ODI} = \mu^d \cdot C_i(h,T) \cdot [dm(\frac{\nu}{\mu})^{d+1} + (m(\frac{\nu}{\mu})^d)^2] \; . \tag{105}$$

In this case, the speedup $\mathbb{S}_{par}$ is

$$\mathbb{S}_{par} = \frac{T_{imp,dir}}{T_{imp,ODI}} = \frac{C_i(h,T) \cdot [dm\nu^{d+1} + (m\nu^d)^2]}{\mu^d \cdot C_i(h,T) \cdot [dm(\frac{\nu}{\mu})^{d+1} + (m(\frac{\nu}{\mu})^d)^2]} \tag{106}$$

$$= \mu \left[1 + \frac{m\nu^{d-1}(1 - \frac{1}{\mu^{d-1}})}{d + m(\frac{\nu}{\mu})^{d-1}} \right] \; ; \tag{107}$$

which, under the reasonable assumption that $m(\frac{\nu}{\mu})^{d-1} >> d$ when $d > 1$, yields an approximate speedup $\mathbb{S}_{par} = \mu^d$.

If we allow for automatic step size selection in the time integrator, we would expect even a greater speedup than predicted above. This is so both because the time integrator would have to adjust only to the dynamics within the specific subdomain in which it works, and because the subproblems would be less stiff than the original problem.

We should also note that implicit methods often are forbiddingly memory demanding when the system is large and modified Newton iterations are used to solve for the stage values. This problem could be diminished by using a 'matrix free' method instead. These are methods that use a Krylov subspace iteration to solve the linear systems arising in the Newton iteration, such that explicit storage of the Jacobian matrices is avoided. In this case preconditioning is crucial (see e.g. [10]).

5 Convergence in the 1D constant coefficient case

Let us now consider the one-dimensional linear hyperbolic initial boundary value problem with constant coefficients

$$u_t + Au_x = f \quad , \quad (x,t) \in (a,b) \times [0,\infty) \tag{108}$$

$$S_+u = w_L \quad , \quad (x,t) \in \{a\} \times [0,\infty) \tag{109}$$

$$S_-u = w_R \quad , \quad (x,t) \in \{b\} \times [0,\infty) \tag{110}$$

$$u = g \quad , \quad (x,t) \in (a,b) \times \{0\} \tag{111}$$

where $-\infty < a < b < \infty$, $A \in \mathbb{R}^{m\times m}$, $u : (x,t) \mapsto u(x,t) \in \mathbb{R}^m$, $f : (x,t) \mapsto f(x,t) \in \mathbb{R}^m$, $w_L : t \mapsto w_L(t) \in \mathbb{R}^{m_1}$, $w_R : t \mapsto w_R(t) \in \mathbb{R}^{m_2}$, $g : x \mapsto g(x) \in \mathbb{R}^m$, and $m_1 + m_2 = m$. We assume that $f \in L^1$ on all straight lines $t = c_1x + c_2$ in $(a,b) \times [0,\infty)$, $c_1 \neq 0$. The boundary conditions are determined by specifying the characteristic variables that enter the domain. If the matrix $S = S(1)$ is decomposed like in (79), corresponding to the m_1 positive eigenvalues and the m_2 negative eigenvalues of A, we see that we get the boundary conditions (109), (110). We have the following result.

THEOREM 5.1 *The ODI algorithm of Section 4, applied to the problem above, yields the exact solution after M iterations.*

Proof. We transform the equations to characteristic form by setting $w = Su$. Let $\tilde{f} = Sf$ and $\tilde{g} = Sg$ and split w into two components $w = (\xi, \eta)^T$ representing wave propagation to the right and to the left, corresponding to the splitting (78) of Λ. If $a_1 = a$, $b_M = b$ and $a_i < b_i = a_{i+1} < b_{i+1}$ for $i = 1, \ldots, (M-1)$ the ODI scheme becomes

$$(\frac{\partial}{\partial t} + \Lambda_+\frac{\partial}{\partial x})\xi_i^{(k+1)} = \tilde{f}_+ \quad , \quad (x,t) \in (a_i,b_i) \times [0,\infty) \tag{112}$$

$$\xi_i^{(k+1)} = \xi_{i-1}^{(k)} \quad , \quad (x,t) \in \{a_i\} \times [0,\infty) \tag{113}$$

$$\xi_i^{(k+1)} = \tilde{g} \quad , \quad (x,t) \in (a_i,b_i) \times \{0\} \tag{114}$$

and

$$(\frac{\partial}{\partial t} + \Lambda_-\frac{\partial}{\partial x})\eta_i^{(k+1)} = \tilde{f}_- \quad , \quad (x,t) \in (a_i,b_i) \times [0,\infty) \tag{115}$$

$$\eta_i^{(k+1)} = \eta_{i+1}^{(k)} \quad , \quad (x,t) \in \{b_i\} \times [0,\infty) \tag{116}$$

$$\eta_i^{(k+1)} = \tilde{g} \quad , \quad (x,t) \in (a_i,b_i) \times \{0\}\ , \tag{117}$$

where $i = 1, \ldots, M$ and $k = 0, 1, 2, \ldots$; with the understanding that $\xi_0^{(k)} = w_L$ and $\eta_{M+1}^{(k)} = w_R$. We make use of the following lemma.

LEMMA 5.2 *The iteration defined by the single equations*

$$(\frac{\partial}{\partial t} + \lambda\frac{\partial}{\partial x})v_i^{(k+1)} = f \quad , \quad (x,t) \in (a_i,b_i) \times [0,\infty) \tag{118}$$

$$v_i^{(k+1)} = v_{i-1}^{(k)} \quad , \quad (x,t) \in \{a_i\} \times [0,\infty) \tag{119}$$

$$v_i^{(k+1)} = g \quad , \quad (x,t) \in (a_i,b_i) \times \{0\}\ , \tag{120}$$

where $v_0^{(k)} = \omega$ and $\lambda > 0$, gives the exact solution v of the problem

$$(\frac{\partial}{\partial t} + \lambda\frac{\partial}{\partial x})v = f \quad , \quad (x,t) \in (a,b) \times [0,\infty) \tag{121}$$

$$v = \omega \quad , \quad (x,t) \in \{a\} \times [0,\infty) \tag{122}$$

$$v = g \quad , \quad (x,t) \in (a,b) \times \{0\} \tag{123}$$

after M iterations; i.e. $v_i^{(k)} = v\,|_{(a_i,b_i)}$ when $k \geq M$.

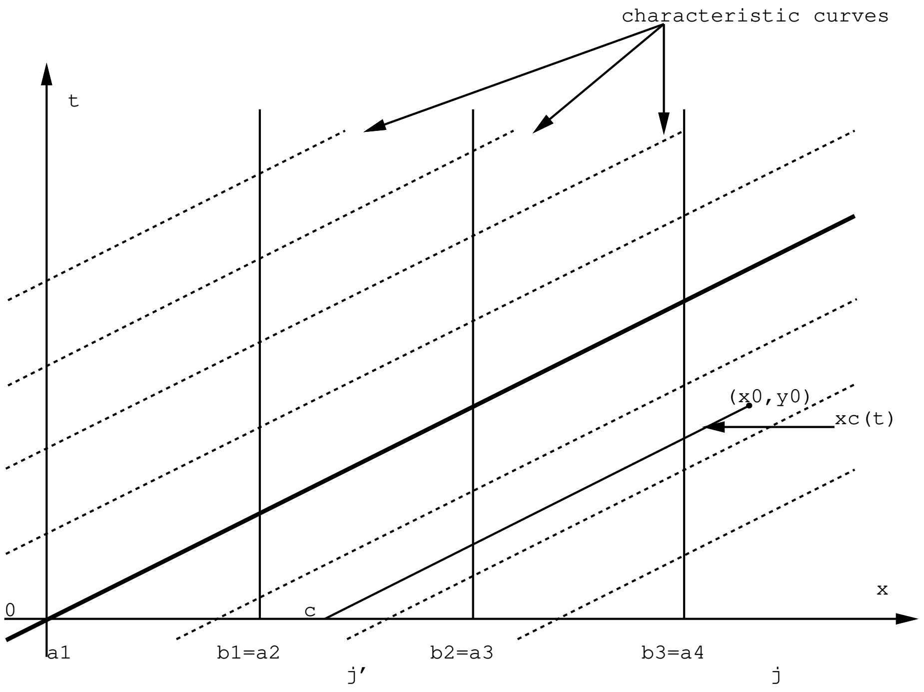

Figure 6: The setting for Case 1 in the proof of the lemma.

Proof. The characteristic curves for the problem is the family of curves given by $dx/dt = \lambda$; i.e. all straight lines of the form $x_c(t) = \lambda t + c$, $c \in \mathbb{R}$. These are also the characteristic curves of the M decomposed problems. Consider a point (x_0, t_0) lying in domain number j, i.e. $a_j < x_0 \leq b_j$. We divide the proof in two different parts.

Case 1. Suppose the characteristic curve $x_c(t)$ through (x_0, t_0) intersects the x-axis in the interval (a, b). In other words, we have that $x_c(0) = c \in (a, b)$. First we note that if $c \in (a_j, b_j)$, we have that

$$v_j^{(k+1)}(x_0, t_0) = g(c) + \int_c^{x_0} f(s, t_c(s))\, ds = v(x_0, t_0)\ ,\ \forall k \geq 0\ ; \tag{124}$$

where $t_c(x)$ is the inverse function of $x_c(t)$. Now, suppose $c \notin (a_j, b_j)$. Then there exists a j' with $0 < j' < j$ s.t. $c \in (a_{j'}, b_{j'})$. See Figure 6. We then have, in iteration $k+1$, that $v_j^{(k+1)}(x_0, t_0)$ is given by

$$v_j^{(k+1)}(x_0, t_0) = v_{j-1}^{(k)}(b_{j-1}, t_c(b_{j-1})) + \int_{a_j}^{x_0} f(s, t_c(s))\, ds\ . \tag{125}$$

Iterating the argument, we get that if $(k - (j - j') + 1) \geq 0$

$$v_j^{(k+1)}(x_0, t_0) = v_{j'}^{(k-(j-j')+1)}(b_{j'}, t_c(b_{j'})) + \int_{a_{j'+1}}^{x_0} f(s, t_c(s))\, ds\ . \tag{126}$$

Using the result above, this implies that

$$v_j^{(k+1)}(x_0, t_0) \quad = \quad g(c) + \int_c^{b_{j'}} f(s, t_c(s))\, ds + \int_{a_{j'+1}}^{x_0} f(s, t_c(s))\, ds \tag{127}$$

$$= g(c) + \int_c^{x_0} f(s, t_c(s))\, ds \quad (128)$$

$$= v(x_0, t_0) \quad (129)$$

when $k - (j - j') \geq 0$. Since $(j - j') \leq M - 1$, this is satisfied for all j if $k \geq M - 1$.

Case 2. If $x_c(t)$ does not intersect the x-axis in (a, b), it clearly intersects the left boundary in the point $(a_1, t_c(a_1))$ where $t_c(a_1) \geq 0$. If $x_0 \in (a_1, b_1)$ we immediately have that

$$v_1^{(k+1)} = \omega(t_c(a_1)) + \int_{a_1}^{x_0} f(s, t_c(s))\, ds = v(x_0, t_0)\ ,\ \forall k \geq 0\ ; \quad (130)$$

Let $x_0 \in (a_j, b_j)$, $j \neq 1$. We then have, in iteration $k+1$, that $v_j^{(k+1)}(x_0, t_0)$ is given by (like above)

$$v_j^{(k+1)}(x_0, t_0) = v_{j-1}^{(k)}(b_{j-1}, t_c(b_{j-1})) + \int_{a_j}^{x_0} f(s, t_c(s))\, ds\ . \quad (131)$$

Again, iterating the argument, we get that if $(k - j + 2) \geq 0$

$$v_j^{(k+1)}(x_0, t_0) = v_1^{(k-j+2)}(b_1, t_c(b_1)) + \int_{a_2}^{x_0} f(s, t_c(s))\, ds\ . \quad (132)$$

Combining this with the result (130) for the leftmost domain, we get

$$v_j^{(k+1)}(x_0, t_0) = \omega(t_c(a_1)) + \int_{a_1}^{b_1} f(s, t_c(s))\, ds + \int_{a_2}^{x_0} f(s, t_c(s))\, ds \quad (133)$$

$$= \omega(t_c(a_1)) + \int_{a_1}^{x_0} f(s, t_c(s))\, ds \quad (134)$$

$$= v(x_0, t_0) \quad (135)$$

when $k - j + 1 \geq 0$. Since $j \leq M$, this is satisfied for all j when $k \geq M - 1$.

As $k = 0$ corresponds to the first iteration, $k = 1$ to the second etc.; we now see that we get the exact solution after M iterations. □

Obviously, we have the same result for the problem in which $\lambda < 0$ and the boundary values are specified on the right boundaries. Since all equations in (112)-(117) are in the form given in the lemma (or its analogue for negative λ), the iteration yields the exact solution after M iterations, and the proof of the theorem is complete. □

Remark 5.3 Note that we did not make any assumptions on the initial iterates $v_j^{(0)}$ in the proof of the lemma. So the theorem remains valid independently of what we choose the initial iterates to be.

Remark 5.4 Taking $s = 3$ in the context of the text following Lemma 4.4, we verify that if $S_+ g = w_L$, $S_+(-Ag_x + f) = \dot{w}_L$ and $S_+(A^2 g_{xx} - Af_x + f_t) = \ddot{w}_L$ for $(x, t) = (a, 0)$, and the same is satisfied for $(x, t) = (b, 0)$, with S_- instead of S_+ and w_R instead of w_L, and in addition $g \in H^3(\Omega)$, $f \in H^3([0, T] \times \Omega)$ and $w_L, w_R \in H^{7/2}([0, T] \times \partial\Omega)$, then u is continuously differentiable with respect to both x and t.

We also have a convergence result in the case where we impose periodic boundary conditions on the problem (108)-(111).

THEOREM 5.5 *The ODI algorithm of Section 4, applied to the problem above, with periodic boundary conditions, will after ℓ iterations yield the exact solution for $t \leq (\ell-1)\cdot d_{\min}/r_\sigma(A)$; where*

$$d_{\min} = \min_{1\leq j\leq M}(b_j - a_j)\ , \tag{136}$$

and $r_\sigma(A)$ is the spectral radius of A.

Proof. The proof goes along the same lines as the proof of Theorem 5.1. However, now the characteristic curves $x_c(t)$ never terminate at a boundary, they rather wrap around and enter on the opposite boundary; always terminating in the interval $[a,b)$. Writing this out, we will see that the solution $v_j^{(k+1)}(x_0,t_0)$ of the problem in Lemma 5.2 now equals the exact solution $v(x_0,t_0)$ if $t_0 \leq (kd_{j-1} + (x_0 - a_j))/\lambda$. A fortiori, this means that $v_j^{(k+1)}(x_0,t_0) = v(x_0,t_0)$ whenever $t_0 \leq kd_{j-1}/\lambda$. For the whole system, this means that $u_j^{(k+1)}(x_0,t_0) = u(x_0,t_0)$ whenever $t_0 \leq kd_{\min}/r_\sigma(A)$, which was the statement of the theorem. □

6 ODE formulation in the ODI case

As mentioned before, we can expect the ODEs coming from the spatial discretization of our PDE to become stiff when the number of nodes, or collocation points, increases. Hence, implicit time integration schemes will be of interest, since they can allow the size of the time steps to be governed by accuracy requirements rather than by stability requirements. In this section, we will see how to formulate the system (97)-(100), discretized by the spectral collocation method, as a pure ODE system (101). In this way we enable the application of any ODE solver. We restrict to the case where the matrix B equals S_-; i.e. where we read characteristic variables on the exterior boundary. It should be possible to apply the ideas of this section to other discretization techniques than the spectral collocation method as well.

The results in this section are applications of the results in [3], where an ODE formulation of quasilinear hyperbolic PDEs in a d-dimensional cube is derived, with boundary conditions of the form $S_- u_t = \varphi$. By an ODE formulation we mean an ODE initial value problem coming from the spatial discretization of a PDE initial-boundary problem; i.e. such that the boundary conditions of the PDE problem are incorporated into the right hand side of the ODE system. In [3] this was accomplished by making use of a differential boundary treatment (see e.g. Kopriva [7] or Quarteroni [19]) together with simple matrix partitionings as (78) and (79). The results are summarized below.

Consider the quasilinear hyperbolic system

$$(\frac{\partial}{\partial t} + L)u = f \quad , \quad (x,t) \in \Omega \times [0,T] \tag{137}$$

$$S_- u_t = \varphi \quad , \quad (x,t) \in \partial\Omega \times [0,T] \tag{138}$$

$$u = g \quad , \quad (x,t) \in \Omega \times \{0\}\ ; \tag{139}$$

where L and $S_- = S_-(\xi_p)$ are as in Section 4, and $\Omega = (-1,1)^d$. To ensure a smooth solution, we should in the linear case impose restrictions on the data according to Lemma 4.4. Now, let $t \geq 0$ and let p be a collocation point. Let $\Lambda = \Lambda(\xi_p)$ and $S = S(\xi_p)$ be partitioned as before, and let $S^{-1} = S^{-1}(\xi_p)$ be partitioned accordingly. We denote by $\mathsf{u}(t)$ the polynomial approximation to $u(t)$ and use the notation

$$\mathsf{u}_p(t) := \mathsf{u}(t)\mid_{x=p} , \tag{140}$$

$$(D_{x_i}\mathsf{u})_p := \frac{\partial}{\partial x_i}\mathsf{u}(t)\mid_{x=p} . \tag{141}$$

Then we have that for $p \in \Omega$, the ODE for u_p becomes by straightforward collocation

$$\dot{\mathsf{u}}_p = f - \sum_{j=1}^{d} A_j(\mathsf{u}_p, p, t)(D_{x_j}\mathsf{u})_p - C(\mathsf{u}_p, p, t)\mathsf{u}_p . \tag{142}$$

However, if $p \in \partial\Omega$ such that the normal vector ξ_p pointing out of $\overline{\Omega}$ in p is defined, the ODE for u_p becomes

$$\dot{\mathsf{u}}_p = S_+^{-1}S_+\left[f - \sum_{j=1}^{d} A_j(\mathsf{u}_p, p, t)(D_{x_j}\mathsf{u})_p - C(\mathsf{u}_p, p, t)\mathsf{u}_p\right] + S_-^{-1}\varphi_p . \tag{143}$$

As remarked in [3], we usually (as opposed to (138)) have a boundary condition of the form

$$S_- u = \tilde{\omega} \quad , \quad (x,t) \in \partial\Omega \times [0,T] ; \tag{144}$$

hence we must translate this into a boundary condition of the form (138) in order to use the results above. Using the chain rule for differentiation, we get

$$S_- u_t = \frac{d}{dt}(S_- u) - \frac{d}{dt}(S_-)u , \tag{145}$$

where $\frac{d}{dt}$ should be understood as a total derivative. Let us take a closer look at the matrix $d/dt(S_-)$. We have that

$$\frac{d}{dt}(S_-) = \frac{\partial}{\partial t}(S_-) + \sum_{i=1}^{m} \frac{\partial}{\partial u_i}(S_-)\frac{\partial}{\partial t}u_i , \tag{146}$$

so if we apply it to a vector z we get

$$\begin{aligned}
\frac{d}{dt}(S_-)z &= \left[\frac{\partial}{\partial t}(S_-) + \sum_{i=1}^{m} \frac{\partial}{\partial u_i}(S_-)\frac{\partial}{\partial t}u_i\right] z & (147)\\
&= \frac{\partial}{\partial t}(S_-)z + \sum_{i=1}^{m} \frac{\partial}{\partial u_i}(S_-)z \cdot \frac{\partial}{\partial t}u_i & (148)\\
&= \frac{\partial}{\partial t}(S_-)z + \mathcal{R}(S_-, z)u_t ; & (149)
\end{aligned}$$

where the matrix $\mathcal{R}(E,z)$ is given by

$$\mathcal{R}(E,z) = \left[\frac{\partial}{\partial u_1}(E)z \ \cdots \ \frac{\partial}{\partial u_m}(E)z\right] , \tag{150}$$

i.e. its columns consists of the m vectors $\frac{\partial}{\partial u_i}(E)z$.

Returning to the ODE formulation, we first consider the case where $\tilde{\omega} = b$, cf. (99). We then get

$$S_- u_t = b_t - \frac{d}{dt}(S_-)u \ , \tag{151}$$

and the resulting ODE for u_p in a point $p \in \partial\overline{\Omega}$ where this type of boundary condition is imposed, becomes

$$\begin{aligned} \dot{\mathsf{u}}_p &= S_+^{-1}S_+ \left[f - \sum_{j=1}^{d} A_j(\mathsf{u}_p, p, t)(D_{x_j}\mathsf{u})_p - C(\mathsf{u}_p, p, t)\mathsf{u}_p \right] \\ &+ S_-^{-1}\left[\dot{b}_p - \frac{\partial}{\partial t}(S_-)\mathsf{u}_p - \mathcal{R}(S_-, \mathsf{u}_p)\dot{\mathsf{u}}_p \right] \ . \end{aligned} \tag{152}$$

The reason for introducing the matrix $\mathcal{R}$ is now apparent, as it makes it possible to get a linear equation for $\dot{\mathsf{u}}_p$ in the form

$$\begin{aligned} \left[I + S_-^{-1}\mathcal{R}(S_-, \mathsf{u}_p)\right]\dot{\mathsf{u}}_p &= S_+^{-1}S_+ \left[f - \sum_{j=1}^{d} A_j(\mathsf{u}_p, p, t)(D_{x_j}\mathsf{u})_p - C(\mathsf{u}_p, p, t)\mathsf{u}_p \right] \\ &+ S_-^{-1}\left[\dot{b}_p - \frac{\partial}{\partial t}(S_-)\mathsf{u}_p \right] \ . \end{aligned} \tag{153}$$

In the same manner, we consider the case in which $\tilde{\omega} = S_-\omega$, cf. (98). We now have that

$$S_- u_t = \frac{d}{dt}(S_-\omega) - \frac{d}{dt}(S_-)u \tag{154}$$

$$= S_-\omega_t - \frac{d}{dt}(S_-)(u - \omega) \ , \tag{155}$$

and the ODE in a point $p \in \partial\overline{\Omega}$ with this kind of boundary condition becomes

$$\begin{aligned} \dot{\mathsf{u}}_p &= S_+^{-1}S_+ \left[f - \sum_{j=1}^{d} A_j(\mathsf{u}_p, p, t)(D_{x_j}\mathsf{u})_p - C(\mathsf{u}_p, p, t)\mathsf{u}_p \right] \\ &+ S_-^{-1}\left[S_-\dot{\omega}_p - \frac{\partial}{\partial t}(S_-)(\mathsf{u}_p - \omega_p) - \mathcal{R}(S_-, \mathsf{u}_p - \omega_p)\dot{\mathsf{u}}_p \right] \ . \end{aligned} \tag{156}$$

Again, after rearranging terms, we get

$$\begin{aligned} \left[I + S_-^{-1}\mathcal{R}(S_-, \mathsf{u}_p - \omega_p)\right]\dot{\mathsf{u}}_p &= S_+^{-1}S_+ \left[f - \sum_{j=1}^{d} A_j(\mathsf{u}_p, p, t)(D_{x_j}\mathsf{u})_p - C(\mathsf{u}_p, p, t)\mathsf{u}_p \right] \\ &+ S_-^{-1}\left[S_-\dot{\omega}_p - \frac{\partial}{\partial t}(S_-)(\mathsf{u}_p - \omega_p) \right] \ . \end{aligned} \tag{157}$$

If L is linear and the coefficient matrices A_j are independent of t, it is clear that the matrices $\mathcal{R}$ and $\partial/\partial t(S_-)$ disappear, and we have explicit expressions for $\dot{\mathsf{u}}_p$. In some cases where the matrices A_j depend on t, and in some quasilinear cases we might approximate our equations by deleting the last term, if we know a priori that S_- does not vary much as t varies. In the general case, however, we can not neglect the influence of this term, and we should solve the equations with respect to $\dot{\mathsf{u}}_p$. In order to get a unique solution of the linear system, we must then have either

$$\det\left[I + S_-^{-1}\mathcal{R}(S_-, \mathsf{u}_p)\right] \neq 0 \tag{158}$$

or

$$\det\left[I + S_-^{-1}\mathcal{R}(S_-, \mathsf{u}_p - \omega_p)\right] \neq 0 \; ; \tag{159}$$

depending on the type of boundary condition we are dealing with.

As an example, consider the one-dimensional Euler equations for compressible adiabatic flow on an open interval $(a, b) \subset \mathbb{R}$, given by the quasilinear system

$$\rho_t + v\rho_x + \rho v_x = 0 \; , \tag{160}$$

$$v_t + v v_x + \frac{1}{\rho} p_x = 0 \; , \tag{161}$$

$$p_t + \gamma p v_x + v p_x = 0 \; . \tag{162}$$

We assume that the flow is subsonic, i.e. that $|v| < c$; where $c = \sqrt{\gamma p/\rho}$ is the speed of sound. At the right boundary we have $p = b$, $\xi_b = 1$, and the matrix $(I + S_-^{-1}(1)\mathcal{R}(S_-(1), \mathsf{u}_b - \omega_b))$ that would arise e.g. if (a, b) was a subdomain in an ODI algorithm, is (for the case of outflow, i.e. $v_b(t) > 0$)

$$\left[I + S_-^{-1}\mathcal{R}(S_-, \mathsf{u}_b - \omega_b)\right] = \begin{bmatrix} 1 + \frac{2\,\omega_2\, c\rho_b - 2\, v_b c\rho_b - \omega_3 + p_b}{4\, c^2 \rho_b} & 0 & -\frac{p_b - \omega_3}{4\, c^2 p_b} \\ -\frac{2\,\omega_2\, c\rho_b - 2\, v_b c\rho_b - \omega_3 + p_b}{4\, \rho_b^2 c} & 1 & \frac{p_b - \omega_3}{4\, c\rho_b\, p_b} \\ \frac{2\,\omega_2\, c\rho_b - 2\, v_b c\rho_b - \omega_3 + p_b}{4\, \rho_b} & 0 & 1 - \frac{p_b - \omega_3}{4\, p_b} \end{bmatrix} . \tag{163}$$

The requirement (159) for unique solution is now

$$\det = \frac{3\, c^2 \rho_b\, p_b + c^2 \rho_b\, \omega_3 + 2\, \omega_2\, c\rho_b\, p_b - 2\, v_b c\rho_b\, p_b - \omega_3\, p_b + p_b^2}{4\, c^2 \rho_b\, p_b} \neq 0 \; . \tag{164}$$

Assuming $|v_b| < Kc$, $|\omega_2| < Kc$, $\omega_3 > 0$ and $\gamma > 1$, where $K \leq 3/4 + 1/(4\gamma)$, we have that (dropping the subscript b for convenience)

$$\begin{aligned} \det &= \frac{1}{4cp}\left[3cp(1 + 1/(3\gamma)) + c\omega_3(1 - 1/\gamma) - 2pv + 2p\omega_2\right] && (165) \\ &> \frac{1}{4cp}\left[3cp(1 + 1/(3\gamma)) + c\omega_3(1 - 1/\gamma) - 4cpK\right] && (166) \\ &= \frac{1}{4cp}\left[3pc(1 - 4K/3 + 1/(3\gamma)) + c\omega_3(1 - 1/\gamma)\right] && (167) \\ &\geq 0 \, , && (168) \end{aligned}$$

since $p, c > 0$; and we can solve for $\dot{\mathsf{u}}_b$ uniquely. For a monoatomic gas $\gamma = 5/3$, and we are assured uniqueness with $K = 0.9$; which normally would be more than sufficient for subsonic simulations.

In our ODI algorithm, ω in (155) will in iteration $k + 1$ in domain i be the solution in the adjacent domain j computed in the previous sweep; i.e. $\omega = \mathsf{u}_{j,p}^{(k)}$. Note that if the collocation points in two neighbouring domains i and j coincide on their common boundary, $\omega(t) = \mathsf{u}_{j,p}^{(k)}(t)$ will be given explicitly, as the polynomial valued function $\mathsf{u}_j^{(k)}(t)$ is represented by its values in the collocation points. If the collocation points do not coincide, we have to form the polynomial explicitly, or use an interpolation formula, in order to get $\mathsf{u}_{j,p}^{(k)}(t)$. A similar problem arises when temporal discretization is included. If we allow for independent time stepping in the different subdomains in every sweep, we would in this case also have to interpolate in the time variable to get the desired input values.

7 Numerical experiments

We have tested the ODI algorithm presented in Section 4 for some one and two-dimensional hyperbolic problems. We used the ODE formulation of the previous section, using Chebyshev spectral collocation for the spatial discretization. The resulting ODEs were then integrated in time by second order Runge-Kutta methods with automatic step size selection. Both explicit and implicit Runge-Kutta methods were used, with absolute and relative error tolerance set to 10^{-4}. All tests were run by the same C program; only the function describing the ODE formulation of the specific problem was changed. The program uses dynamic memory allocation and linked lists in order to store the boundaries during each sweep. After each sweep, adjacent domains switch the specific boundary lists, and linear interpolation in the lists is used to read boundary values during the next sweep. For the parallel computations we used the PVM software [1] on a cluster of 4 Sun IPX workstations. The sequential tests were timed with the `user` part of the unix `/bin/time` command, while the parallel tests were timed with the `real` part. It should be emphasized that the parallel tests were run with normal priority on a multi-user local network, implying the possibility of some time-sharing effects.

7.1 1D constant coefficient test problem

We solve the following system

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t + \begin{bmatrix} \frac{1}{2} & 1 \\ 1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_x = 0 \tag{169}$$

for $(x,t) \in (-1,1) \times [0,2]$, with initial conditions

$$u_1(x,0) = 0 \ , \ u_2(x,0) = \sin \pi x \tag{170}$$

and boundary conditions

$$S_+(1)u(-1,t) = \sin \frac{3\pi}{2} t = w_L \ , \ S_-(1)u(1,t) = -\sin \frac{\pi}{2} t = w_R \ ; \tag{171}$$

which is of the type studied in Section 5. We have that $A(\xi) = A \cdot \xi$ and

$$S(\xi) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \ , \ \Lambda(\xi) = \begin{bmatrix} \frac{3}{2}\xi & 0 \\ 0 & -\frac{1}{2}\xi \end{bmatrix} \ . \tag{172}$$

Table 1: Timings for 1D test problem ($M = \mu = 4$)

	Explicit RK method		Implicit RK method	
Collocation points (N)	60	120	60	120
Direct method	6.0	105.3	33.4	136.6
ODI, sequential	9.4	36.8	77.2	262.0
ODI, parallel (PVM)	4.8	12.1	25.3	100.3
Real speedup (PVM)	1.3	8.7	1.3	1.4

Table 2: Timings for 1D Euler equations ($M = \mu = 4$)

	Explicit RK method		Implicit RK method	
Collocation points (N)	60	120	60	120
Direct method	7.9	142.4	115.5	1739.1
ODI, sequential	10.2	50.8	108.3	391.8
ODI, parallel (PVM)	5.6	19.3	44.4	165.9
Real speedup (PVM)	1.4	7.4	2.6	10.5

The initial and boundary conditions satisfy the compatibility conditions given in Remark 5.3, and hence the solution is continuously differentiable. The timing results are given in Table 1 for the single domain calculation, the ODI method run sequentially, and the ODI method run in parallel. We get some speedup, varying from 1.3 to 8.7; which is not in very good correspondence with the idealized estimate in Section 4. Plots showing the four sweeps and the error decay are given in Figure 7, where both the supnorm and the l^2-norm in $\Omega \times [0, T]$ of the difference between the direct solution and the ODI solution are plotted. It is easy to see how the ODI method behaves from the plots; information flows across more and more subdomains as the number of sweeps increases. As we proved above, the ODI algorithm only needs four iterations to converge in this case.

7.2 1D Euler equations

The one-dimensional Euler equations for adiabatic compressible flow are given by (160)-(162), and we look for their solution for $(x, t) \in (-1, 1) \times [0, 2]$. We have the initial conditions

$$\rho(x, 0) = 1\,, \tag{173}$$

$$v(x, 0) = 0.05\,, \tag{174}$$

$$p(x, 0) = \begin{cases} \exp(1/4 - (x - 1/4)^2) & , \quad -1/4 < x < 3/4 \\ 1 & , \quad \text{else} \end{cases} \tag{175}$$

and simulate open boundaries by prescribing characteristic combinations of the steady state solution for the incoming characteristics; i.e. we have $\tilde{\omega} = S_{-}\omega$ in (144), where $\omega \equiv (1, 0.05, 1)^T$. The eigenvalues of $A(\xi)$ are $\lambda_1 = (v - c) \cdot \xi_1$, $\lambda_2 = v \cdot \xi_1$ and $\lambda_3 = (v + c) \cdot \xi_1$. We assume subsonic flow, i.e. that $|v| < c$. The timing results for the different combinations of methods and problem sizes are given in Table 2. The speedup in this case varies from 1.4 to 10.5, and the numbers indicate that the speedup increases with the size of the problem.

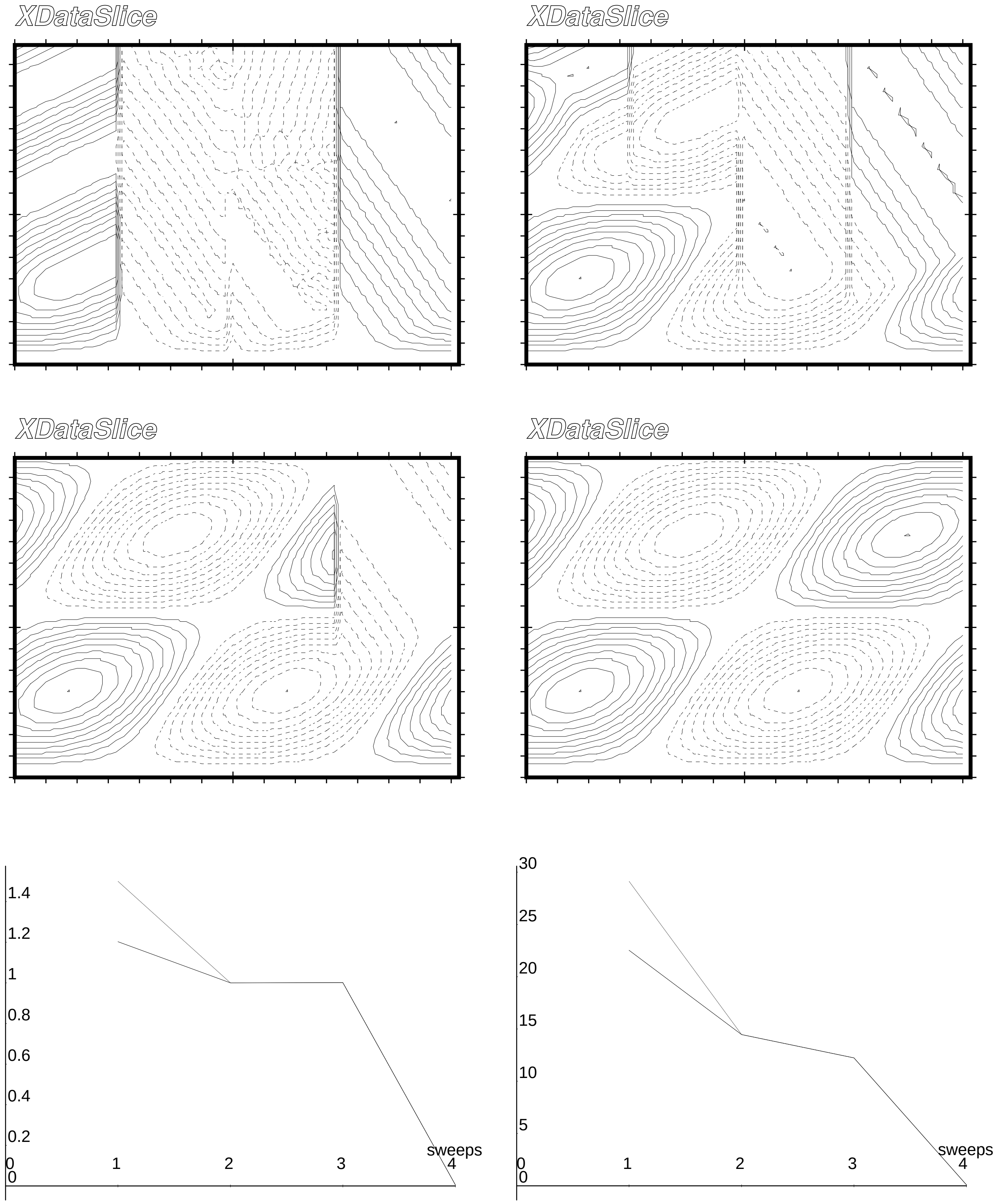

Figure 7: The 4 sweeps of the ODI method for the 1D test problem: $u_1(x,t)$. $N = \nu = 60, M = \mu = 4$. Errors in supnorm and l^2-norm.

Table 3: Timings for 2D test problem ($M = \mu^2 = 4$)

	Explicit RK method		Implicit RK method	
Collocation points ($N = \nu^2$)	900	3600	900	3600
Direct method	36.7	930.8	20615.4	(*)
ODI, sequential	77.4	1284.8	(**)	(**)
ODI, parallel (PVM)	30.5	369.5	2616.5	(**)
Real speedup (PVM)	1.2	2.5	7.9	

Contour plots of the sweeps are given in Figure 8, together with plots of the two errors. In this case we have no convergence results, but the plots indicate that four sweeps give reasonable results.

7.3 2D linear test problem

As a simple two-dimensional hyperbolic problem, we consider the system

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t + \begin{bmatrix} \frac{1}{2} & 1 \\ 1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_x + \begin{bmatrix} \frac{5}{8} & \frac{3}{8} \\ \frac{3}{8} & \frac{5}{8} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_y = 0 \tag{176}$$

for $(x, y, t) \in (-1, 1) \times (-1, 1) \times [0, 1.5]$. We give as initial conditions the functions

$$u_1(x, y, 0) = 0.05 \exp(-30(x^2 + y^2)) , \tag{177}$$

$$u_2(x, y, 0) = 0.05 \exp(-30(x^2 + (y - 0.2)^2)) , \tag{178}$$

and simulate open boundaries as in the previous example. The coefficient matrices in this system are simultaneously diagonalizable, and the eigenvalues of $A(\xi)$ are $\lambda_1 = -1/2 \cdot \xi_1 + 1/4 \cdot \xi_2$ and $\lambda_2 = 3/2 \cdot \xi_1 + \xi_2$. For this problem we put $M = 4$ and $\mu = 2$; i.e. we divided the square into 4 subsquares. Four sweeps were computed by the ODI method. In the corner points, we chose ξ_p as the unit vector pointing out of the square, oriented along its diagonal. In the interior corner point, all four subdomains are neighbours. In this point, we chose to read the boundary value from the neighbouring subdomain in the horizontal direction only, for simplicity. This is not believed to give a stable treatment of the corners in the general case. The timing results, comparing the direct method with the ODI method, are given in Table 3. The symbol (*) means that the program aborted due to insufficient memory on the machine, while (**) means that the tests were not made. Snapshots of the solution at $t = 1.0$ in the ODI approach are plotted in Figure 9, together with error plots in space and time. From the error plots, it would seem that three sweeps are sufficient in this case.

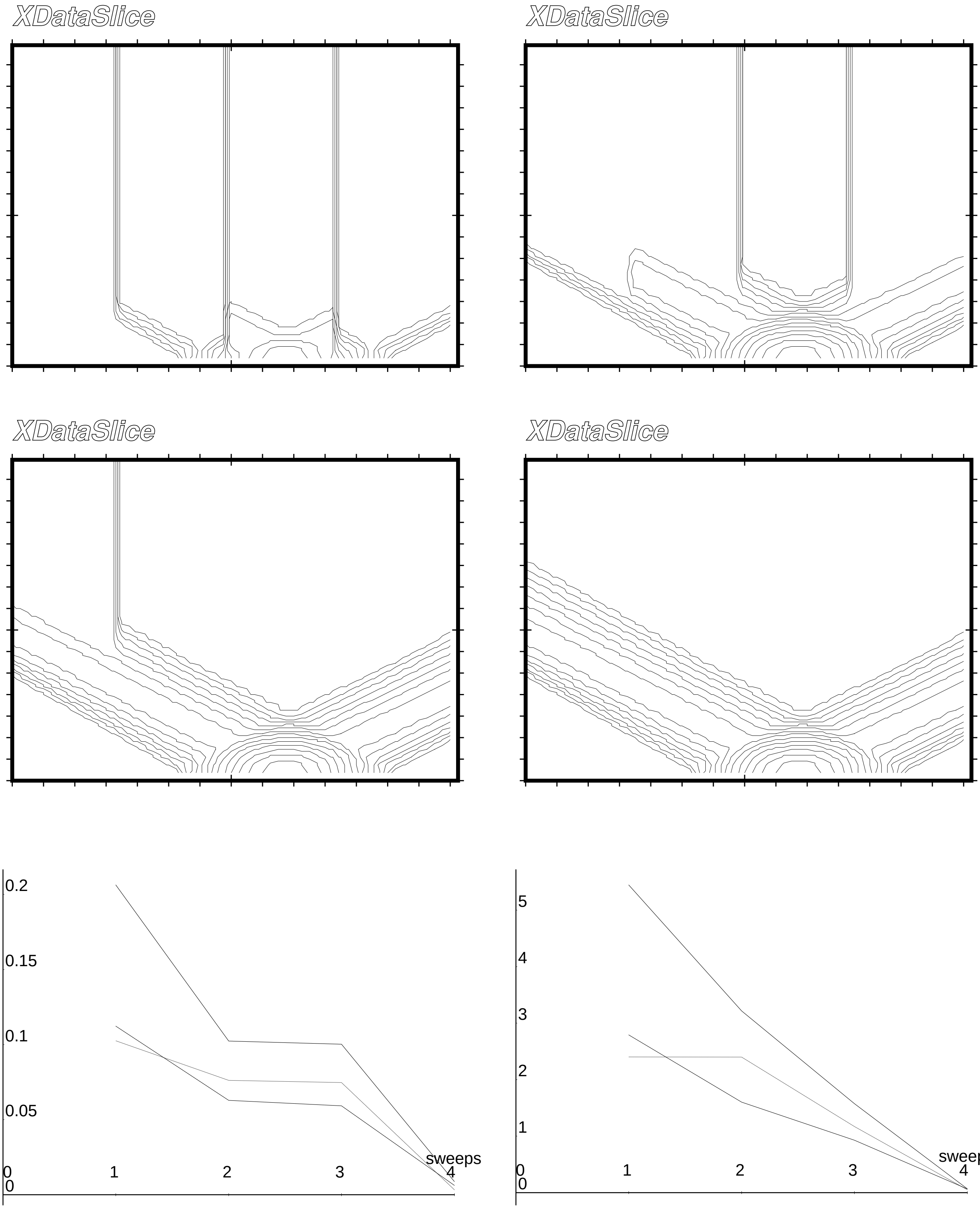

Figure 8: The 4 sweeps of the ODI method for the 1D Euler equations: $p(x,t)$. $N = \nu = 60, M = \mu = 4$. Errors in supnorm and l^2-norm.

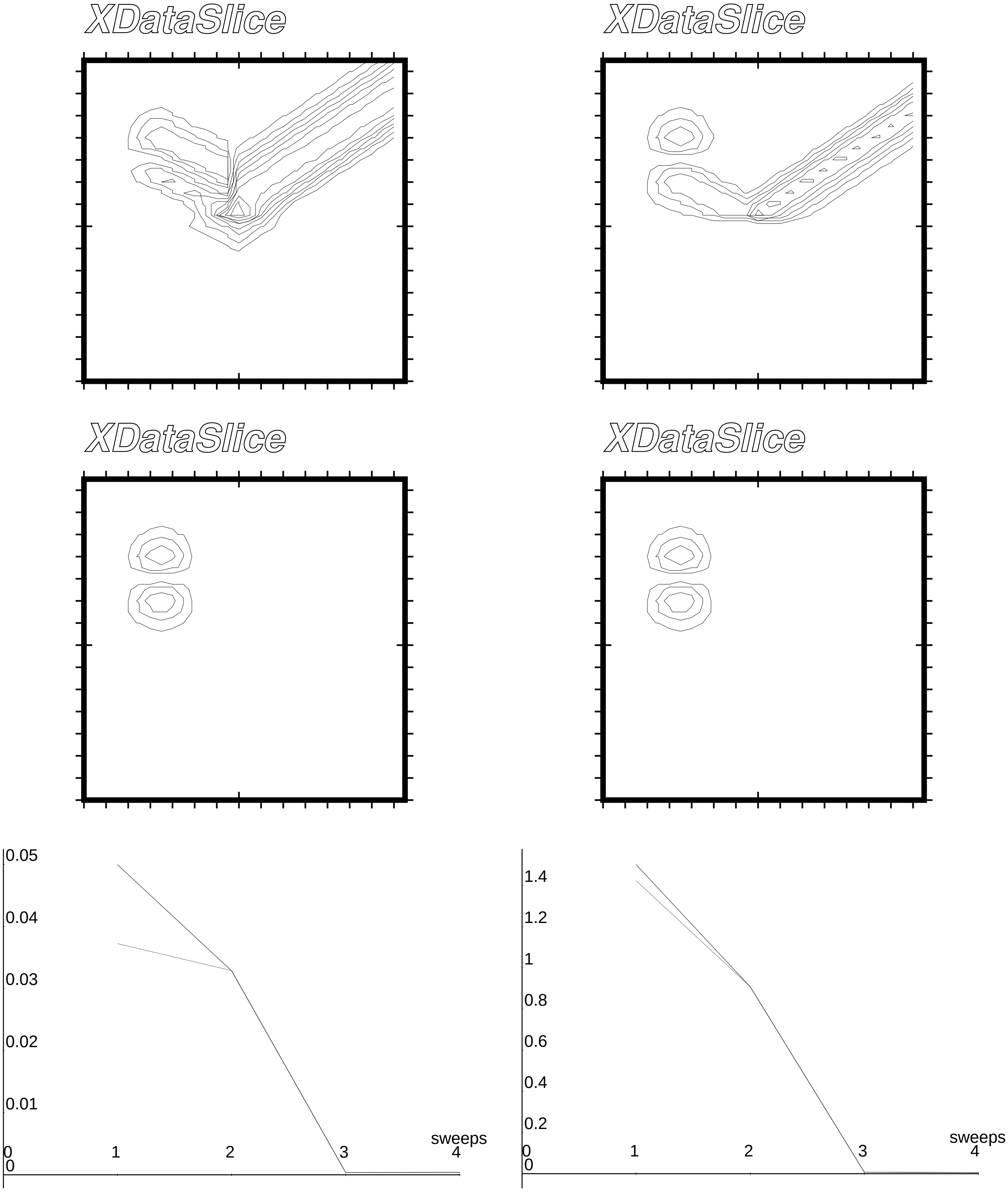

Figure 9: The 4 sweeps of the ODI method for the 2D test problem: $u_1(x,t)$. $N = \nu^2 = 900, M = \mu^2 = 4$. Plots of the solution at $t = 1.0$. Errors in supnorm and l^2-norm.

8 Concluding remarks

In this report, we have extended the idea of dynamic iteration to the PDE case through a domain decomposition approach. The resulting algorithm, called the ODI algorithm, seems especially suitable for computations on parallel computers. It is expected that the ODI algorithm will be competitive to direct methods when integrating large systems on parallel computers; both with respect to time consumption and memory requirements. This is indeed indicated by the numerical experiments performed.

We have in particular concentrated our study to ODI for hyperbolic PDEs, and convergence results are given for the one-dimensional constant coefficient case. However, the convergence properties of ODI for higher dimensional problems should also be examined, and this will be subject to future research. When it comes to ODI for parabolic PDEs, we have no results yet.

References

[1] Adam Beguelin, Jack Dongarra, Al Geist, Robert Manchek, and Vaidy Sunderam, *A users' guide to PVM parallel virtual machine*, Tech. Report ORNL/TM-11826, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1991.

[2] Morten Bjørhus, *Dynamic iteration for second order linear initial value problems*, Master's thesis, Division of Mathematical Sciences, Norwegian Institute of Technology, Trondheim, Norway, 1990.

[3] ______, *The ODE formulation of hyperbolic PDEs discretized by the spectral collocation method*, Submitted to SIAM J. Sci. Stat. Comput., December 1991.

[4] Claudio Canuto, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang, *Spectral methods in fluid dynamics*, Springer-Verlag, Berlin Heidelberg, 1988.

[5] Ernst Hairer, Syvert Paul Nørsett, and Gerhard Wanner, *Solving ordinary differential equations I*, Springer-Verlag, Berlin Heidelberg, 1987.

[6] Alan Jeffrey, *Quasilinear hyperbolic systems and waves*, Pitman Publishing, London, 1976.

[7] David A. Kopriva, *Computation of hyperbolic equations on complicated domains with patched and overset chebyshev grids*, SIAM J. Sci. Stat. Comput. **10** (1989), 120–132.

[8] Benedict Leimkuhler, *Waveform relaxation for multibody dynamics*, Tech. Report Research Report A278, Helsinki University of Technology, Institute of Mathematics, Helsinki, 1990.

[9] Ekachai Lelarasmee, Albert E. Ruehli, and Alberto L. Sangiovanni-Vincentelli, *The waveform relaxation method for time-domain analysis of large scale integrated circuits*, IEEE Trans. on CAD of IC and Syst. **1** (1982), 131–145.

[10] Ivar Lie, *Using implicit ODE methods with iterative linear equation solvers in spectral methods*, To appear in SIAM J. Sci. Stat. Comput.

[11] Ernest Lindelöf, *Sur l'application des méthodes d'approximations successives à l'etude des intégrales réelles des équations différentielles ordinaires*, Journal de mathématiques pures et appliquées **10** (1894), 117–128.

[12] Ulla Miekkala and Olavi Nevanlinna, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Stat. Comput. **8** (1987), 459–482.

[13] Olavi Nevanlinna, *Remarks on Picard-Lindelöf iteration*, BIT **29** (1989), 328–346, 535–562.

[14] ______, *Linear acceleration of Picard-Lindelöf iteration*, Numer. Math. **57** (1990), 147–156.

[15] ______, *Power bounded prolongations and Picard-Lindelöf iteration*, Numer. Math. **58** (1990), 479–501.

[16] Émile Picard, *Sur l'application des méthodes d'approximations successives à l'etude de certaines équations différentielles ordinaires*, Journal de mathématiques pures et appliquées **9** (1893), 217–271.

[17] Phoolan Prasad and Renuka Ravindran, *Partial differential equations*, Wiley Eastern, New Dehli, 1985.

[18] Alfio Quarteroni, *Domain decomposition method for the numerical solution of partial differential equations*, Tech. Report Research Report UMSI 90/246, University of Minnesota Supercomputer Institute, Minneapolis, Minnesota, 1990.

[19] ______, *Domain decomposition methods for systems of conservation laws: Spectral collocation approximations*, SIAM J. Sci. Stat. Comput. **11** (1990), 1029–1052.

[20] Jeffrey B. Rauch and Frank J. Massey III, *Differentiability of solutions to hyperbolic initial-boundary value problems*, Trans. Am. Math. Soc. **189** (1974), 303–318.

[21] Pierre-Arnaud Raviart and Jean-Marie Thomas, *Introduction à l'analyse numérique des équations aux dérivées partielles*, Masson, Paris, 1988.

[22] Roar Skålin, *Numerisk løsning av ordinære differensialligninger ved parallellisering på systemnivå. et studium av Runge-Kutta metoder*, Master's thesis, Division of Mathematical Sciences, Norwegian Institute of Technology, Trondheim, Norway, 1988.

[23] Jacob White, Alberto Sangiovanni-Vincentelli, Farouk Odeh, and Albert E. Ruehli, *Waveform relaxation: Theory and practice*, Trans. Soc. Comp. Sim. **2** (1985), 95–133.

[24] Marino Zennaro, Personal communication, June 1992.