Parallelization in time through tensor-product space-time solvers

Yvon Maday[1], Einar M. Rønquist[2]

1 - Université Pierre et Marie Curie-Paris6, UMR 7598, Laboratoire J.-L. Lions, Paris, F-75005 France and Division of Applied Mathematics, Brown University, 182 George Street, Providence, RI 02912, USA

2 - Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway

Abstract

In this note, we extend the fast tensor-product algorithm for the simulation of time-dependent partial differential equations. We use the natural tensorization of the space-time domain to propose, after discretization, a set of independent problems, each one with the complexity of a single steady problem. This allows for an efficient parallel implementation that is already interesting on small architectures, but that can also be combined with standard domain-decomposition-based algorithms providing a further direction of parallelism on large computer platforms. Preliminary numerical simulations are presented for a one-dimensional unsteady heat equation.

Résumé

Dans cette note on généralise à la simulation de phénomènes instationnaires l'algorithme de produit tensoriel. On utilise la tensorisation naturelle du domaine espace-temps pour proposer, après discrétisation un ensemble de problèmes indépendants, chacun d'eux ayant la complexité d'un simple problème stationnaire. Ceci permet une mise en œuvre parallèle déjà interessante sur des petites architectures mais elle peut être également combinée avec des techniques classiques de décomposition de domaine pour utiliser au mieux des architectures avec un nombre de processeurs plus important. Des premiers résultats sont présentés sur un problème de la chaleur instationnaire monodimensionnel.

Version française abrégée

La résolution numérique de problèmes modélisés par des équations aux dérivées partielles est maintenant assez bien établie pour des phénomènes stationnaires. Les techniques itératives comme le "multishooting" de [5], les approches multigrilles de [2], et plus récemment les approches pararéelles en temps de [3] permettent de traiter des phénomènes instationnaires de façon efficace en tenant compte des architectures des calculateurs parallèles. La méthode non itérative, que nous présentons ici est une généralisation des techniques de tensorisation présentées par exemple dans [1] pour des discrétisations en espace seulement ;

Email addresses: maday@ann.jussieu.fr (Yvon Maday[1]), ronquist@math.ntnu.no (Einar M. Rønquist[2]).

Preprint submitted to Elsevier Science

elle peut être combinée à des techniques classiques de décomposition de domaine en espace pour une plus grande efficacité.

Considérons par exemple l'équation de la chaleur (1, 2, 3). Après discrétisation en espace par une méthode classique, par exemple de différences finies de pas $\Delta x = L/(N+1)$ et une discrétisation en temps implicite, on obtient le système discret (5) où le vecteur $\underline{u}^m \in \mathbb{R}^N$ (resp. $\underline{f}^m \in \mathbb{R}^N$) représente l'approximation de la solution (resp. la donnée de flux de chaleur) aux nœuds internes et au temps t^m, et $\Delta t_m = t^m - t^{m-1}$ est le pas de temps au niveau t^m. La matrice $\underline{A} \in \mathbb{R}^{N\times N}$ est classiquement symétrique définie positive.

À chaque pas de temps on doit donc inverser l'opérateur de Helmholtz discret $\underline{A} + \frac{1}{\Delta t_m}\underline{I}$ et la détermination de $\underline{u}^{m-1}$ est nécessaire pour calculer $\underline{u}^m$. Pour briser la nature intrinsèquement séquentielle de cette résolution, on utilise l'algorithme de produit tensoriel rapide. Soit $\underline{A} \in \mathbb{R}^{N\times N}$ et $\underline{B} \in \mathbb{R}^{M\times M}$; on note tout d'abord $\underline{C} = \underline{B} \otimes \underline{A}$ le produit tensoriel de B et A : c'est la matrice $\underline{C} \in \mathbb{R}^{(MN)\times(MN)}$ de coefficients $c_{ij} = b_{kl}a_{mn}$, $1 \le k,l \le M$, $1 \le m,n \le N$, $i = m + (k-1)N$ et $j = n + (l-1)N$. Avec ces notations, le système discret (5) s'écrit (8) où $\underline{B}$ est défini en (6), $\underline{I}_t$ est la matrice identité de dimension M et $\underline{I}_x$ est la matrice identité de dimension N, de plus $\underline{u} \in \mathbb{R}^{MN}$ (resp. $\underline{f} \in \mathbb{R}^{MN}$) représente le vecteur concaténé de la solution et du membre de droite selon (7) à tous les pas de temps. Si l'on suppose tous les Δt_m, $m = 1,\dots,M$ différents, la matrice $\underline{B}$ est diagonalisable, selon (9). Le système (8) peut donc prendre la forme (13) où l'inconnue $\underline{w}$ est définie en (12) et le membre de droite $\underline{g}$ est défini en (15). On remarque aisément que ce système se déploie en M problèmes indépendants (14) qui peuvent donc être résolus en même temps sur des processeurs parallèles, et la solution $\{\underline{u}^m, m = 1,\dots,M\}$ s'obtient par la transformation inverse (16).

En notant par W_H la complexité de résolution d'un système de type (5), la complexité de résolution de l'approche tensorisée (non parallélisée) est donnée en (17), en effet la diagonalisation de la matrice $\underline{B}$ compte pour $\mathcal{O}(M^2)$ et le calcul des membres de droite (15) ainsi que la transformation inverse (16) compte pour $\mathcal{O}(M^2N)$. Si l'on dispose par contre de $P = M$ processeurs parallèles, la complexité par processeur devient (18), en ajoutant une complexité de communication entre les processeurs W_{comm} on obtient une efficacité de parallélisme S donnée par (19). Ainsi si l'on suppose d'une part que $W_{\text{comm}} \ll W_H$ et en outre $M \ll N$ et $MN \ll W_H$ on conclut à une efficité quasi optimale $S \approx M$.

La tensorisation exposée ci dessus peut être effectuée pour des schémas en temps plus précis que le schémas implicite d'ordre 1 de Euler, tant que la matrice $\underline{B}$ reste triangulaire inférieure. La simulation numérique suivante basée sur un schémas implicite rétrograde d'ordre 3 (avec un choix de pas de temps $\Delta t_k = \rho^{k-1}\Delta t_1$, $k > 1$, avec $\rho = 1.2$ se sorte que $\Delta t_{15} \approx 13\Delta t_1$, et $\Delta T = \sum_{k=1}^{M=15} \Delta t_k = 0.592$). L'erreur obtenue après un temps T égal à $T = 100\Delta T = 59.2$ est $1.4 \cdot 10^{-3}$ alors qu'elle est de $4.3 \cdot 10^{-2}$ avec les schémas d'ordre 1.

1. Introduction

The simulation of three-dimensional steady problems governed by partial differential equations is nowadays quite common; the size of current computers allows one to get very good approximations, close to experimental data, for a large class of problems. The availability of parallel computers has resulted in many contributions related to the parallelization of problems with respect to the spatial directions. A key ingredient has been to divide the global problem through domain decomposition, thus allowing smaller problems to be assigned to different processors.

The next challenge, which has already started to be faced, is to solve time-dependent problems with the same efficiency and ease. Roughly speaking, this amounts to adding an additional direction to the three spatial ones. Most discretizations of time-dependent problems rely on using finite difference methods

in time. This results in the need to solve a sequence of stationary-like problems, for which domain decomposition is primarily the only way to exploit parallel computers. Indeed, the sequence of these stationary-like problems is recursive and is not obviously broken into independent subproblems. Among the emerging new methods, let us quote the *multishooting technique* introduced by Nievergelt [5], and the *multigrid approach* introduced by Hackbusch [2]. More recently a new approach, named the *parareal in time method* has been introduced by Lions, Maday and Turinici [3] with various contributions showing the interest of this predictor-corrector methodology.

The approach we present here is of a completely different nature. First of all it is not iterative, at least in the version we propose here to tackle linear problems. Second, it may be of interest already with very few processors. Third, as a consequence of the diversity of available solvers, it can be combined with the parareal approach as we shall explain in the full paper associated with this Note. The method used here is purely algebraic. It is inspired by the fast tensor product solver that we introduced in the recent paper [1] for spatial discretizations, and has also some connection with the classical way that leads to the harmonic version of a time dependent problem, and that looks for solutions of the form $e^{i\omega t}u(x)$. Finally, let us indicate that it can easily be combined with standard domain decomposition methods in space, thus offering enhanced overall parallelism.

2. The heat equation

Consider the one-dimensional unsteady heat equation

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2} + f(x,t) \qquad \text{in } \omega_x = (0,L), \tag{1}$$

subject to the homogeneous boundary conditions

$$u(x=0,t) = u(x=L,t) = 0 \qquad \forall\, t \in \omega_t = [0,T], \tag{2}$$

and to the initial condition

$$u(x,t=0) = g(x). \tag{3}$$

Here, $u(x,t)$ is the solution (the temperature), κ is the thermal diffusivity, and $f(x,t)$ is a thermal heat source that depends on space and time. We assume that the thermal diffusivity, κ, is a constant.

As an introductory example to the approach we will present, we consider the numerical solution of this problem using finite difference methods in space. The spatial domain ω_x is discretized using a uniform grid with grid spacing $\Delta x = L/(N+1)$. Using a standard second order finite difference formula for the diffusion term, we end up with a system of ordinary differential equations of the form

$$\frac{d\underline{u}(t)}{dt} + \underline{A}\,\underline{u}(t) = \underline{f}(t). \tag{4}$$

Here, the vector $\underline{u} \in \mathbb{R}^N$ represents the numerical solution at the N internal grid points (at a particular time t), the vector $\underline{f} \in \mathbb{R}^N$ represents the given data at the internal grid points (at a particular time t), and the matrix $\underline{A} \in \mathbb{R}^{N\times N}$, that represents the discrete diffusion operator, is symmetric positive definite.

Assume now that we discretize the system of ordinary differential equations using an implicit method like the first order backward Euler scheme. The set of fully discrete equations can then be expressed as

$$\frac{\underline{u}^m - \underline{u}^{m-1}}{\Delta t_m} + \underline{A}\,\underline{u}^m = \underline{f}^m \quad , \quad m = 1, \ldots M, \tag{5}$$

where $\underline{u}^m$ denotes the numerical approximation of the solution at the internal grid points at time t^m and Δt_m is the time step at time level t^m; more precisely, $\Delta t_m = t^m - t^{m-1}$.

This simple approach to obtain the numerical solution results in the need to solve a system of algebraic equations at each time step, with the discrete Helmholtz operator $\underline{A} + \frac{1}{\Delta t_m}\underline{I}$ as the system matrix. Note that this is indeed a purely sequential approach; we need to know $\underline{u}^{m-1}$ before we can compute $\underline{u}^m$ by solving a discrete Helmholtz problem.

3. Tensor-product representation

We will now present a reformulation of the model problem introduced in the previous section. This reformulation is based on a tensor product representation of time and space and will allow us to obtain a scheme with parallel processing possibilities in the time domain.

First, let us briefly review the tensor product notation that we will be using. Let $\underline{A} \in \mathbb{R}^{N\times N}$ and $\underline{B} \in \mathbb{R}^{M\times M}$. Then the tensor product of $\underline{B}$ and $\underline{A}$, $\underline{C} = \underline{B} \otimes \underline{A}$, gives a matrix $\underline{C} \in \mathbb{R}^{(MN)\times(MN)}$ with matrix entrees $c_{ij} = b_{kl}a_{mn}$, with $1 \leq k, l \leq M$, $1 \leq m, n \leq N$, $i = m + (k-1)N$ and $j = n + (l-1)N$.

Associated with the backward Euler scheme (5), we define the matrix $\underline{B}$ to be

$$\underline{B} = \begin{pmatrix} \frac{1}{\Delta t_1} & & & & \\ -\frac{1}{\Delta t_2} & \frac{1}{\Delta t_2} & & & \\ & -\frac{1}{\Delta t_3} & \frac{1}{\Delta t_3} & & \\ & & & \ddots & \\ & & & -\frac{1}{\Delta t_M} & \frac{1}{\Delta t_M} \end{pmatrix} . \tag{6}$$

We also define $\underline{I}_t$ to be the identity matrix of dimension M (associated with the time domain) and $\underline{I}_x$ to be the identity matrix of dimension N (associated with the spatial domain). Finally, we define the concatenated solution vector $\underline{u} \in \mathbb{R}^{MN}$ and the concatenated right hand side $\underline{f} \in \mathbb{R}^{MN}$ as

$$\underline{u} = \begin{pmatrix} \underline{u}^1 \\ \underline{u}^2 \\ \underline{u}^3 \\ . \\ \underline{u}^M \end{pmatrix}, \qquad \underline{f} = \begin{pmatrix} \underline{f}^1 \\ \underline{f}^2 \\ \underline{f}^3 \\ . \\ \underline{f}^M \end{pmatrix} . \tag{7}$$

With this notation we can write the (5) in tensor product notation as

$$(\underline{B} \otimes \underline{I}_x + \underline{I}_t \otimes \underline{A})\ \underline{u} = (\underline{I}_t \otimes \underline{I}_x)\ \underline{f}\ . \tag{8}$$

Note that this system of algebraic equations represents the numerical solution at M different time levels. Although the system matrix is sparse, it does not a priori seem like a good strategy to reduce the overall computational cost.

In order to see the potential benefit of this approach, we proceed as follows. We first assume that it is possible to diagonalize the matrix $\underline{B}$, i.e., we assume that we can write $\underline{B}$ as

$$\underline{B} = \underline{S}\Lambda\underline{S}^{-1}. \tag{9}$$

Here, the columns of $\underline{S}$ are the eigenvectors of $\underline{B}$, while Λ is a diagonal matrix comprising the corresponding eigenvalues of $\underline{B}$. For our particular Euler backward scheme, the matrix $\underline{B}$ is lower triangular, and the eigenvalues are equal to the diagonal entries of $\underline{B}$. Hence, the matrix $\underline{B}$ is diagonalizable only if all the time steps Δt_m, $m = 1, \ldots, M$ are *different*. Assuming that this is the case, we can write (8) as

$$(\underline{S}\Lambda\underline{S}^{-1} \otimes \underline{I}_x + \underline{I}_t \otimes \underline{A})\ \underline{u} = (\underline{I}_t \otimes \underline{I}_x)\ \underline{f}\ , \tag{10}$$

or

$$(\underline{S} \otimes \underline{I}_x)\ (\Lambda \otimes \underline{I}_x + \underline{I}_t \otimes \underline{A})\ (\underline{S}^{-1} \otimes \underline{I}_x)\ \underline{u} = (\underline{I}_t \otimes \underline{I}_x)\ \underline{f}. \tag{11}$$

Introducing

$$\underline{w} = (\underline{S}^{-1} \otimes \underline{I}_x)\ \underline{u} \tag{12}$$

and premultiplying (11) by $(\underline{S}^{-1} \otimes \underline{I}_x)$ gives

$$(\Lambda \otimes \underline{I}_x + \underline{I}_t \otimes \underline{A})\ \underline{w} = (\underline{S}^{-1} \otimes \underline{I}_x)\ \underline{f}. \tag{13}$$

An equivalent formulation of (13) is thus

$$(\underline{A} + \lambda_m \underline{I}_x)\ \underline{w}^m = \underline{g}^m, \quad m = 1, \ldots, M, \tag{14}$$

with a concatenated right hand side

$$\underline{g} = (\underline{S}^{-1} \otimes \underline{I}_x)\ \underline{f}. \tag{15}$$

We have here reformulated the solution in terms of M computational steps. However, in contrast to the systems (5) which represent M *sequential* steps, the systems (14) represent M *independent* steps. Hence, (14) can be solved in parallel in a completely decoupled fashion without any communication.

Once the systems (14) have been solved for $\underline{w}^m$, $m = 1, \ldots, M$, we compute $\underline{u}^m$, $m = 1, \ldots, M$ from (12), i.e.,

$$\underline{u} = (\underline{S} \otimes \underline{I}_x)\ \underline{w}. \tag{16}$$

4. Computational complexity

We now discuss the computational complexity associated with the approach presented in the previous section. We recall that the model problem is a simple unsteady heat transfer problem discretized with a standard finite difference method in space and a backward Euler scheme in time. The extension to use finite element methods in space is straightforward.

The computational approach assumes that it is possible to diagonalize the matrix $\underline{B}$ defined in (6). The cost of the diagonalization (9) is $\mathcal{O}(M^2)$ where M is the number of (standard) time steps following a backward Euler approach. The corresponding memory requirement is $\mathcal{O}(M^2)$.

Next, the cost of computing the transformed right hand sides (15) is $\mathcal{O}(M^2 N)$, while the cost of solving the M systems in (14) is the same as the standard approach for solving the M systems (5); the particular cost here depends on the chosen solver for these Helmholtz-type systems. Let us denote the cost per Helmholtz system as W_H. The cost of performing the final transformation (16) is $\mathcal{O}(M^2 N)$.

The total (serial) cost W_1 of the new approach is therefore

$$W_1 = c_1 \cdot M^2 + c_2 \cdot M^2\, N + M \cdot W_H. \tag{17}$$

The cost $\widetilde{W_1}$ of following a standard approach is just the last term on the right hand side.

However, assuming now the we have available $P = M$ processors, the computational complexity per processor is

$$W_P = c_1 \cdot M^2 + c_2 \cdot M\, N + W_H. \tag{18}$$

In addition, we have a communication cost, W_{comm}, associated with collecting the concatenated vector $\underline{w}$ on each processor (all-to-all communication), which is needed in order to compute $\underline{u}^m$ on each processor P_m, $m = 1, \ldots, M$; see (16). The speedup S is thus

$$S = \frac{\widetilde{W}_1}{W_P + W_{\text{comm}}} = \frac{M \cdot W_H}{c_1 \cdot M^2 + c_2 \cdot M\, N + W_H + W_{\text{comm}}}. \tag{19}$$

If $W_{\text{comm}} \ll W_H$, $M \ll N$, and the work W_H satisfies $MN \ll W_H$, we see that $S \approx M$, i.e., we get close to perfect speedup. The assumption $W_{\text{comm}} \ll W_H$ can be expected to be satisfied when considering the extension of this approach to solve more realistic multi-dimensional problems. In this case, the assumption $M \ll N$ can also easily be fulfilled, and the assumption $MN \ll W_H$ follows in most cases.

5. Generalization and numerical results

The method we have just proposed is first order in time, and since it requires that all the time steps are different, the accuracy will be related to the largest time step. In order to make the method more efficient, we propose to use a higher order scheme in time with time steps $\Delta t_k = \rho^{k-1}\Delta t_1$, $k > 1$, with ρ larger but close to 1, e.g. $\rho = 1.2$ so that $\Delta t_{15} \approx 13\Delta t_1$, and $\Delta T = \sum_{k=1}^{M=15} \Delta t_k = 0.592$. Note that, as can be expected, choosing ρ much closer to 1 may lead to instabilities due to numerical errors. The scheme we use here comprises a second order Crank-Nicolson scheme for the first time step, followed by a Backward Difference scheme of order 2 to get $\underline{u}^2$, then a Backward Difference scheme of order 3 to get $\underline{u}^k$ for $2 < k \leq M = 15$. The scheme is a little bit more involved but results in a matrix similar to $\underline{B}$ defined in (6) since $\underline{B}$ is still lower diagonal, with few terms (actually 3) under the diagonal, and the same diagonal terms as in (6). The complexity is identical as for the first order version we analyzed. We have performed the simulation for a right hand side that has an explicit exact solution $u(x,t) = \sin(\pi t)\sin(\pi x)$. The error at time $T = 100\Delta T = 59.2$ is $1.4 \cdot 10^{-3}$ with the "high" order scheme compared with $4.3 \cdot 10^{-2}$ for the first order scheme. A pure spectral Galerkin method was here used for the spatial discretization.

A further generalization, where we extend to the fullest the notion of a high order method, can be proposed. It consists of proposing a spectral method in time, and we refer to [4] for a first implementation that requires one to work in the complex field.

Acknowledgements. Part of this work has been done in the frame of the grant ANR-06-CIS6-007.

References

[1] T. Bjøntegaard, Y. Maday, E.M. Rønquist Fast tensor-product solvers. Part I: Partially deformed three-dimensional domains, submitted 2007 and http://www.ann.jussieu.fr/publications/2007/R07023.html.

[2] W. Hackbusch, Parabolic multi-grid methods, in Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski and J.-L. Lions, eds., North-Holland, 1984, pp. 189–197.

[3] J.-L. Lions, Y. Maday, and G. Turinici, A parareal in time discretization of pde's, C.R. Acad. Sci. Paris, Serie I, 332 (2001), pp. 661–668.

[4] Y. Maday, E.M. Rønquist Fast tensor-product solvers: Part II: Spectral discretization in space and time, submitted 2007 and http://www.ann.jussieu.fr/publications/2007.

[5] J. Nievergelt, Parallel methods for integration ordinary differential equations, Comm. ACM, 7 (1964), pp. 731–733.