Computers & Fluids 52 (2011) 22–32

Contents lists available at SciVerse ScienceDirect

Computers & Fluids

journal homepage: www.elsevier.com/locate/compfluid

A fast tensor-product solver for incompressible fluid flow in partially deformed three-dimensional domains: Parallel implementation

Arne Morten Kvarving, Einar M. Rønquist *

Norwegian University of Science and Technology, Department of Mathematical Sciences, Trondheim, Norway

ARTICLE INFO

Article history:
Received 27 September 2010
Received in revised form 20 May 2011
Accepted 9 August 2011
Available online 30 August 2011

Keywords:
Tensor-product solver
Shared memory
Distributed memory
Bénard–Marangoni

ABSTRACT

We describe a parallel implementation of the tensor-product solver derived in Refs. [6,19]. A combined distributed/shared memory model is chosen, since the flexibility allows us to map the algorithm better to the available resources. Since the approach requires special attention to load balancing, we also propose a scheme that resolves the challenges involved. Speedup results from test problems, as well as from real simulations, are presented and discussed. While the speedups are not perfect, we show that the new algorithms are more than competitive with a standard 3D approach parallelized using domain decomposition.

1. Introduction

Parallelization strategies for finite element codes are usually based on the domain decomposition paradigm [13,30]. This way of extracting parallelism has many attractive features, such as the availability of very efficient solvers, relatively easy implementation and access to load balanced codes. In principle, all that is required is a sufficient number of elements compared to the number of processors, as well as a good division of these elements between the processors. This gives a coarse grained division of the workload which maps very well to a distributed memory model where separate processes communicate through message passing.

In this document we study parallelization strategies for another class of algorithms, which allows for alternative approaches. These algorithms are only applicable to a certain class of problems, namely problems in geometries of the "cylindrical" kind which can be viewed as an extrusion of some general 2D cross-section [6,11,19]; see Fig. 1 for an example. Our motivation for considering this particular class of geometries is that we want to use the code to simulate surface-tension-driven Bénard–Marangoni convection in confined containers [5,7,8,17,18,27,29]. This class of algorithms decompose the elliptic 3D problems into a set of completely decoupled 2D problems. The decoupling into several subproblems offers a new parallelization strategy; instead of having all processors participate in the solution of one large problem, we can now divide them into groups which work independently of each other. In other words, the parallelization strategy is to a large extent given by the algorithm. The division of the processors into groups means that we have to face additional challenges when it comes to load balancing, since the different groups may have varying solution times. Nonetheless, these algorithms are interesting since from experience [6,19] they reduce the number of floating point operations needed by close to an order in magnitude and are very conservative with respect to memory usage. When applicable, they should reduce the amount of computing resources needed significantly.

In terms of spatial discretization, the focus in this work is on spectral elements. We remark this does not reflect a limitation of the algorithms considered. The algorithms can be used with any spatial discretization, such as low-order finite elements or finite differences. None of the following discussion relies directly on the use of spectral elements, thus the parallelization derived is also applicable with these kinds of spatial discretizations. For instance, earlier work has been reported where a Fourier expansion in the extrusion direction is combined with an element method [9,12,16,21]. Our approach can certainly be applied in this case.

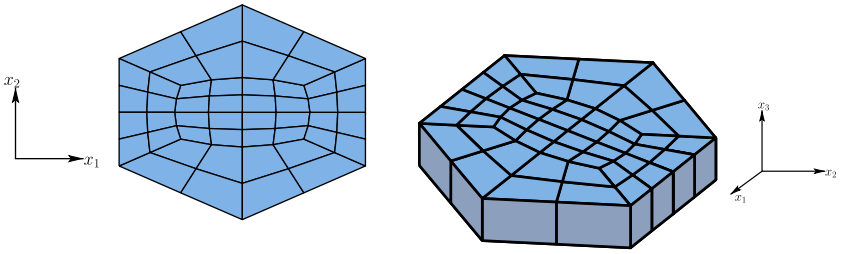
The outline of the paper is as follows. In Section 2 the description of the (model) problem considered is given. In Section 3 we briefly discuss the discretization leading to the linear systems of equations we have to solve. In Section 4 we give a brief overview of the tensor-product solvers considered, before moving on to discuss the parallelization in Section 5. Speedup results are then given in Section 6. Finally, in Section 7, we summarize our findings and present our conclusions.

2. Problem definition

As a model problem, we consider the unsteady Stokes equations in some (extruded) domain Ω;

* Corresponding author.
E-mail address: ronquist@math.ntnu.no (E.M. Rønquist).

0045-7930/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2011.08.007



(a) cross-section (b) 3D geometry

Fig. 1. An example of the class of geometries we consider. Here, the cross-section is a hexagon which is extruded in the third direction to form a full 3D container.

$$\begin{aligned} &\frac{\partial \mathbf{u}}{\partial t} - \nabla^2 \mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega, \\ &\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega, \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the velocity, p is the pressure and $\mathbf{f}$ represents a given body force. We assume that appropriate initial and boundary conditions for the velocity are specified. For simplicity, we here assume that

$$\mathbf{u} = \mathbf{0} \quad \text{on } \partial\Omega.$$

The governing equations for incompressible fluid flows are the Navier–Stokes equations. However, since the convection operator is typically handled using explicit time integrators (following a semi-implicit approach), this does not give rise to additional elliptic systems. Thus, the Stokes equations serve us well, since we here consider the solution of the elliptic systems of equations derived from an implicit-in-time method in combination with a velocity–pressure splitting scheme.

3. Discretization

We use high order spectral elements [24] to discretize in space, specifically the $\mathbb{P}_N/\mathbb{P}_{N-2}$ method [26]. The domain Ω is decomposed into K spectral elements, each with polynomial degree N. These elements are layered in the extrusion direction, thus K can be expressed in the form $K = \mathcal{K} \cdot \mathcal{L}$, where $\mathcal{K}$ is the number of spectral elements in each layer, and $\mathcal{L}$ is the total number of layers. We refer to a specific element using two indices: $k_i^j, i = 1, \ldots, \mathcal{K},\ j = 1, \ldots, \mathcal{L}$. Here i is the element number within each layer and j the layer. These elements give rise to a number of planes, one per degree of freedom in the extrusion direction. The number of such planes depends on the boundary conditions considered, as well as N, the polynomial order of the elements. For the homogenous Dirichlet boundary conditions considered here, each velocity component will have $\mathcal{N}_1 = \mathcal{L}N - 1$ planes. Likewise, for the pressure we have $\mathcal{N}_2 = \mathcal{L}(N-1)$ planes; this would be the same no matter which boundary conditions are enforced on the velocity.

In the following, certain operations will take place across the entire span of the extrusion direction. We thus introduce a set of "super-elements", $\mathcal{E}_i = \left\{k_i^j\right\}_{j=1}^{\mathcal{L}}, i = 1, \ldots, \mathcal{K}$. Each super-element $\mathcal{E}_i$ consists of the composition of $\mathcal{L}$ spectral elements in the extrusion direction, i.e., we have $\mathcal{K}$ such super-elements.

The system of semi-discrete equations corresponding to (1) (discrete in space and continuous in time) can be expressed as

$$\begin{aligned} &\mathbf{B}\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} + \mathbf{A}\mathbf{u} - \mathbf{D}^T p = \mathbf{B}\mathbf{f} \\ &\mathbf{D}\mathbf{u} = 0. \end{aligned} \tag{2}$$

Here, $\mathbf{A}$ is the discrete viscous operator (vector Laplacian), $\mathbf{B}$ the mass matrix, $\mathbf{D}$ and $\mathbf{D}^T$ represent the discrete divergence and gradient operator, respectively, while $\mathbf{u}$ and p are the unknown, nodal velocity and pressure values, respectively. Note that we use the same notation for the velocity, pressure and source in (1) and (2). In the following the symbols refer to the discrete quantities.

For clarity of presentation we here consider a first order temporal discretization. We employ a velocity–pressure splitting scheme, specifically an incremental pressure-correction scheme [10,15,31,28]. This is done to avoid a costly coupled solution strategy, as well as the nested iterations associated with a Uzawa decoupling [4,23]. These splitting schemes are closely coupled to the temporal discretization; in particular, they are based on backward differencing. This means that for a first order realization, we use the backward Euler method and our fully discrete problem reads

$$\begin{aligned} &\frac{1}{\Delta t}\mathbf{B}(\hat{\mathbf{u}}^{n+1} - \mathbf{u}^n) + \mathbf{A}\hat{\mathbf{u}}^{n+1} - \mathbf{D}^T p^n = \mathbf{B}\mathbf{f}^{n+1}, \\ &\frac{1}{\Delta t}(\mathbf{u}^{n+1} - \hat{\mathbf{u}}^{n+1}) - \mathbf{D}^T(p^{n+1} - p^n) = 0, \\ &\mathbf{D}\mathbf{u}^{n+1} = 0, \end{aligned} \tag{3}$$

resulting in decoupled problems for the velocity and the pressure

$$\mathbf{H}\hat{\mathbf{u}}^{n+1} = \frac{1}{\Delta t}\mathbf{B}\mathbf{u}^n + \mathbf{D}^T p^n + \mathbf{B}\mathbf{f}^{n+1}, \tag{4}$$

$$\underbrace{\mathbf{D}\mathbf{B}^{-1}\mathbf{D}^T}_{=\mathbf{E}}\Delta p^{n+1} = \mathbf{E}\Delta p^{n+1} = -\frac{1}{\Delta t}\mathbf{D}\hat{\mathbf{u}}^{n+1}. \tag{5}$$

Here Δt is the time step, superscript n refers to the quantity evaluated at time $t^n = n\Delta t$, $\mathbf{H} = \mathbf{A} + \frac{1}{\Delta t}\mathbf{B}$ is the discrete Helmholtz operator, $\mathbf{E}$ is the consistent Poisson operator and $\Delta p^{n+1} = p^{n+1} - p^n$. After we have solved these equations, the nodal values for the pressure and velocity are updated through

$$\begin{aligned} &\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{n+1} + \Delta t\mathbf{B}^{-1}\mathbf{D}^T\Delta p^{n+1}, \\ &p^{n+1} = p^n + \Delta p^{n+1}. \end{aligned} \tag{6}$$

4. Tensor-product algorithms

The key observation behind the new algorithms is the fact that the extruded geometries lead to tensor-product forms for the elliptic operators (Helmholtz and consistent Poisson). Specifically, we have

$$\mathbf{H} = \mathbf{B}^{1D} \otimes \mathbf{A}^{2D} + \mathbf{A}^{1D} \otimes \mathbf{B}^{2D} + \frac{1}{\Delta t}\mathbf{B}^{1D} \otimes \mathbf{B}^{2D}.$$

Here the superscripts $1D$ and $2D$ refer to the one-dimensional and the two-dimensional operators, respectively, i.e., all the geometry deformations are contained in the operators with the $2D$ superscript. Likewise, the consistent pressure operator can be expressed as

$$\mathbf{E} = \mathbf{B}_*^{1D} \otimes \mathbf{E}^{2D} + \mathbf{E}^{1D} \otimes \mathbf{B}_*^{2D},$$

where