NORGES TEKNISK-NATURVITENSKAPELIGE
UNIVERSITET

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

by

Arne Morten Kvarving, Tormod Bjøntegaard, Einar M. Rønquist*

PREPRINT
NUMERICS NO. 9/2010

NORWEGIAN UNIVERSITY OF
SCIENCE AND TECHNOLOGY
TRONDHEIM, NORWAY

This report has URL
http://www.math.ntnu.no/preprint/numerics/2010/N9-2010.pdf
Address: Department of Mathematical Sciences, Norwegian University of Science and Technology, N-7491 Trondheim, Norway.

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

Arne Morten Kvarving, Tormod Bjøntegaard, Einar M. Rønquist*

October 7, 2010

We present a fast solution method for solving partial differential equations in a particular class of three-dimensional geometries; the two-dimensional cross-section can have a general shape, but is assumed to be invariant with respect to the third direction. For each elliptic operator considered here, the approach involves solving a one-dimensional eigenvalue problem, performing two simple transformation steps (matrix-matrix multiplications), and the solution of a set of completely decoupled two-dimensional problems. The focus of this paper is on the extension of the method proposed in [1] to the numerical solution of the unsteady Navier-Stokes equations. Particular attention is given to the solution of the consistent pressure Poisson problem resulting from the use of spectral elements in combination with a semi-implicit approach. Numerical results are presented for selected three-dimensional test problems. A reduction in computational time with a factor of 4-20 compared to alternative approaches has been achieved.

Keywords: tensor-product solver, preconditioning, fluid flow

1 Introduction

We discuss the numerical solution of partial differential equations in a particular class of three-dimensional geometries; the two-dimensional cross-section (in the x_1x_2-plane) can have a general shape, but is assumed to be invariant with respect to the third direction; see Figures 1 and 2. Earlier work has exploited such geometries by approximating the solution as a truncated Fourier series in the x_3-direction [3, 5, 12]. The use of Fourier series puts a severe limitation of the applicability of the method; in particular, only periodic boundary conditions can be imposed. In a recent paper [1] a new solution algorithm was proposed which also exploits the tensor-product feature between the x_1x_2-plane and the x_3-direction. This new algorithm is not limited to periodic boundary conditions, but works for general Dirichlet and Neumann type of boundary conditions. The algorithm also works for problems with variable coefficients as long as these can be expressed as separable functions with respect to the variation in the x_1x_2-plane and the variation in the x_3-direction. The work presented in [1] focuses on solving the standard Poisson problem and the Helmholtz problem. It was mentioned that the new solver could be extended to other problems as well. The present work represents one such extension, namely to the numerical solution

of the unsteady (Navier-)Stokes equations. In this work the spatial discretization is based on finite elements, more specifically, high order spectral elements where the velocity is continuous across element boundaries, while the pressure is discontinuous. A pressure-velocity splitting scheme gives rise to a set of elliptic systems at each time step: one Helmholtz problem for each velocity component and a consistent pressure Poisson problem. The discrete pressure operator is different from the standard Laplace operator and requires special treatment both in terms of being able to use the approach proposed in [1], as well as in terms of preconditioning of the resulting independent two-dimensional systems. In fact, the extension of the approach proposed in [1] to new operators requires special attention in each case, and may not always be possible. The focus of this paper is on the solution of the consistent pressure Poisson problem. The new tensor-product algorithm allows us to solve a 3D problem through the solution of several decoupled 2D subproblems. The algorithm is thus highly parallel in nature. Additionally, in the context of iterative solution strategies, optimal preconditioners are often easier to construct in 2D and are often more efficient (lower constants) compared to their 3D realizations.

The outline of the paper is as follows. In Section 2 we state our model problem and define the class of geometries we consider. The discretization of this problem is then discussed in Section 3. The tensor-product form of the discrete operators is highlighted in Section 4, and this is used to construct the new tensor-product solver for the consistent pressure operator in Section 5. We then move on to consider preconditioning of the resulting systems of equations in Section 6 in the context of iterative solution of the independent two-dimensional systems. In Section 7, numerical results are presented both for fully tensor-product geometries as well as for extruded geometries. In Section 8, we also give some results where we have used the new solver as a preconditioner in fully deformed domains. Finally, in Section 9, we summarize our findings and present our conclusions.

2 The unsteady Stokes problem

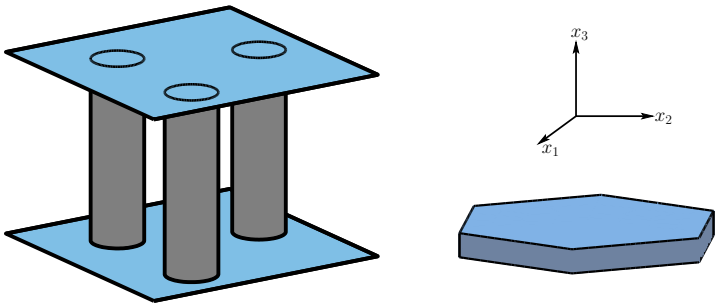
As a model problem we consider the incompressible Stokes equations in a three-dimensional domain Ω,

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

where $\mathbf{u}$ is the velocity, p the pressure and $\mathbf{f}$ a given body force. We assume homogenous Dirichlet boundary conditions for the velocity, however, this assumption will be relaxed and commented on later. We consider geometries which are extrusions of some general 2D cross-sections, i.e.,

$$\Omega = \mathcal{O} \times (0, d),$$

where $\mathcal{O}$ refers to the (general) two-dimensional cross-section, while d is the extrusion length in the x_3-direction; see Figure 1 for two examples of such geometries.



(a) Three parallel cylinders (b) A hexagonal container

Figure 1: Two examples of extruded geometries. The left figure depicts a geometry suitable for studying flow past cylinders mounted between two parallel plates. The right figure depicts a geometry suitable for simulating internal flow in a hexagonal container.

3 Discretization

Before we derive the new tensor-product solver we first describe the discretization process leading to the linear systems of equations we have to solve. Our discretization is based on the weak formulation of the unsteady Stokes problem (1). In addition, a pressure-velocity splitting scheme is utilized, specifically an incremental pressure-correction projection scheme [4, 8, 11, 24, 25, 28].

The splitting scheme is given for a solution which is discrete in time, i.e., it relies on the particular temporal discretization used. For simplicity of presentation we here consider a first order BDF scheme; we briefly comment on the extension to higher order temporal schemes later. We introduce the time step Δt and two temporary quantities: an approximate velocity field, $\hat{\mathbf{u}}^{n+1}$, and a pressure extrapolant, $\hat{p}$, both quantities associated with the new time level t^{n+1}. The idea is to first calculate a (in general) non-solenoidal velocity field based on an extrapolant of the pressure at earlier time levels, and then correct this velocity field by projecting it onto the space of solenoidal fields. The pressure extrapolant $\hat{p}$ is taken as one order less than the temporal accuracy of the scheme, i.e., for a first order scheme we consider a zeroth order extrapolant $\hat{p} = p^n$.

If we introduce the function spaces $X = (H_0^1(\Omega))^3$ and $Y = L_0^2(\Omega)$, the weak, semi-discrete problem (discrete in time) can be stated as:

Find $\left(\mathbf{u}^{n+1} \in X, p^{n+1} \in Y\right), n = 0, 1, \cdots$, such that

$$\begin{aligned}
\left(\frac{\hat{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t}, \mathbf{v}\right) + \left(\nabla \hat{\mathbf{u}}^{n+1}, \nabla \mathbf{v}\right) - \left(\hat{p}, \nabla \cdot \mathbf{v}\right) &= \left(\mathbf{f}, \mathbf{v}\right) \quad \forall\ \mathbf{v} \in X, \\
\left(\frac{\mathbf{u}^{n+1} - \hat{\mathbf{u}}^{n+1}}{\Delta t}, \mathbf{v}\right) &= \left(p^{n+1} - \hat{p}, \nabla \cdot \mathbf{v}\right) \quad \forall\ \mathbf{v} \in X, \\
\left(q, \nabla \cdot \mathbf{u}^{n+1}\right) &= 0 \quad \forall\ q \in Y,
\end{aligned}$$

where $(\cdot, \cdot)$ denote the usual L^2 inner product.

At any time level, we thus seek the solution for all the velocity components in the same space, reflecting the fact that all components are subject to the same boundary conditions; this assumption will be relaxed later.

The discretization in space will be based on spectral elements, however, we could also have used other types of finite elements. We decompose the domain into K (spectral) elements, and introduce K affine mappings Φ^k, $k = 1, \ldots, K$, which map the reference domain into these elements, i.e.,

$$\begin{aligned}\Omega &= \bigcup_{k=1}^{K} \Omega^k, \\ \Omega^k &= \mathcal{O}^k \times (0, d), \\ \hat{\Omega} &= (-1, 1)^3, \\ \Phi^k &: \hat{\Omega} \to \Omega^k,\ k = 1, \ldots, K.\end{aligned}$$

In Figure 2 we show some sample geometries together with their elemental decompositions. We now seek our solution in the discrete, polynomial subspaces ([2, 19, 20])

$$\begin{aligned}X_N &= \left\{ \mathbf{v} \in X, \mathbf{v}\left(\mathbf{x}; t\right) \circ \Phi^k \in \left(\mathbb{P}_N \left(\hat{\Omega} \right) \right)^3 \right\}, \\ Y_N &= \left\{ v \in Y, v\left(\mathbf{x}; t\right) \circ \Phi^k \in \mathbb{P}_{N-2} \left(\hat{\Omega} \right) \right\}.\end{aligned}$$

This leads to the discrete problem:

Find $\left(\mathbf{u}^{n+1} \in X_N, p^{n+1} \in Y_N\right), n = 0, 1, \cdots$, such that

$$\begin{aligned}\left(\frac{\hat{\mathbf{u}}^{n+1} - \mathbf{u}^n}{\Delta t}, \mathbf{v} \right) + \left(\nabla \hat{\mathbf{u}}^{n+1}, \nabla \mathbf{v} \right) - \left(\hat{p}, \nabla \cdot \mathbf{v} \right) &= \left(\mathbf{f}, \mathbf{v} \right) \quad \forall\ \mathbf{v} \in X_N, \\ \left(\frac{\mathbf{u}^{n+1} - \hat{\mathbf{u}}^{n+1}}{\Delta t}, \mathbf{v} \right) &= \left(p^{n+1} - \hat{p}, \nabla \cdot \mathbf{v} \right) \quad \forall\ \mathbf{v} \in X_N, \\ \left(q, \nabla \cdot \mathbf{u}^{n+1} \right) &= 0 \quad \forall\ q \in Y_N.\end{aligned}$$

Note that we use the same symbols in both the semi-discrete (discrete in time) and the fully discrete case. In the following, the symbols will always refer to the fully discrete case.

The class of geometries considered allows us to use a basis where each individual basis function can be viewed as the product of some general two dimensional function and a basis function in the x_3-direction, i.e., a three-dimensional basis function $\gamma(x_1, x_2, x_3)$ can be given on the form

$$\gamma\left(x_1, x_2, x_3\right) = \phi\left(x_1, x_2\right) \psi\left(x_3\right).$$

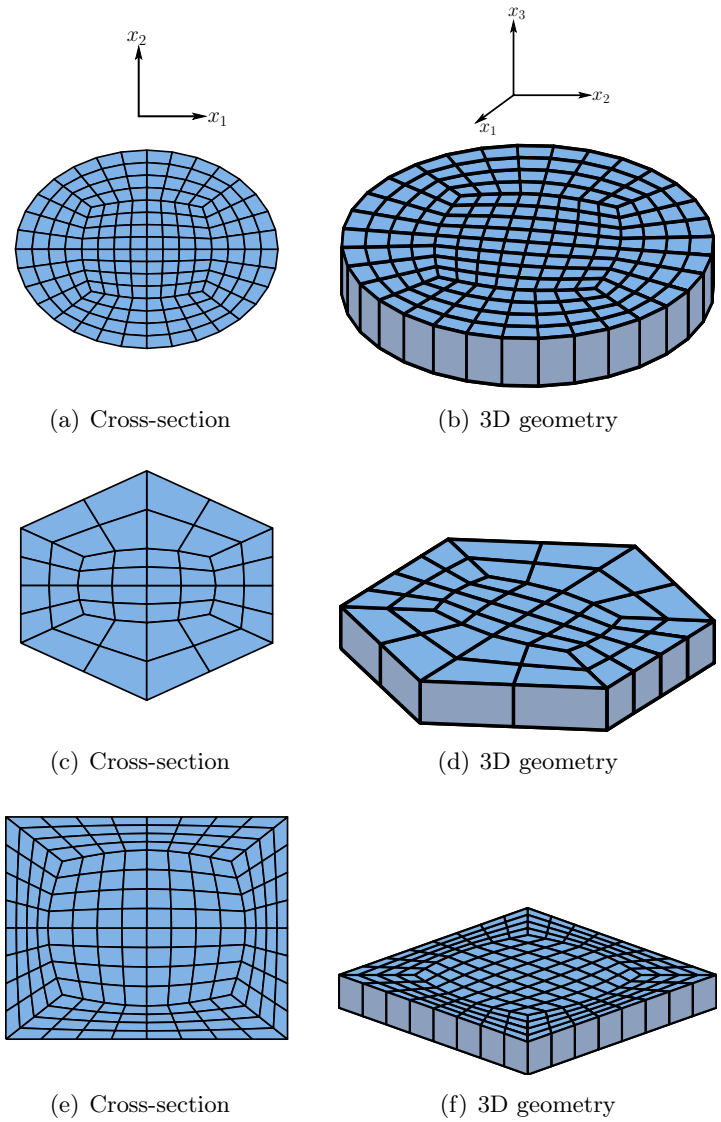
It is a well-known fact that the tensor-product nature of such bases can be exploited to construct fast methods in simple domains. In the next section, we show how this structure can be exploited to construct fast methods suitable for fluid flow problems in the considered class of geometries.

For each velocity component and for each coordinate, we consider a basis which is based on Lagrangian interpolants through the Gauss-Lobatto-Legendre nodes on the reference cube. For example, within each 2D elemental cross-section $\mathcal{O}^k$, we approximate the physical coordinates using a nodal tensor-product basis,

$$x_i^k(\xi, \eta) = \sum_{m=0}^{N} \sum_{n=0}^{N} \left(x_i^k \right)_{mn} \ell_m(\xi) \ell_n(\eta), \qquad i = 1, 2,$$

where ξ and η are the coordinates in the two-dimensional reference domain and $\ell_m(\xi), \ell_n(\eta) \in \mathbb{P}_N(-1, 1)$ are one-dimensional Lagrangian interpolants through the Gauss-Lobatto Legendre nodes. These are combined with the appropriate basis functions in the x_3-direction

(affinely mapped Lagrangian interpolants through the Gauss-Lobatto Legendre points) to form the full basis. The geometry representation is isoparametric and precalculated using a Gordon-Hall mapping procedure [9]. Each velocity component is expressed using a similar basis, while the pressure space uses a nodal basis based on a Gauss-Legendre point distribution on the reference domain. All integrals are evaluated using Gauss quadrature [19].



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

(c) Cross-section (d) 3D geometry

(e) Cross-section (f) 3D geometry

Figure 2: Some examples of the type of geometries we consider together with their elemental decompositions. The cross-sections are depicted in the left column. These are extruded to form the full 3D geometries depicted in the right column.

The set of algebraic equations can then be expressed as

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

where $\mathbf{B}$ is the (velocity) mass matrix, $\mathbf{D}$ is the discrete divergence operator, and $\mathbf{D}^T$ is the discrete gradient operator. The $\mathbf{H}$ operator is the discrete Helmholtz operator,

$$\mathbf{H} = \mathbf{A} + \alpha\mathbf{B},$$

where $\mathbf{A}$ is the discrete Laplace operator and $\alpha = \frac{1}{\Delta t}$ for a first order temporal realization. We now apply block Gaussian elimination to the last two of these systems to end up with the pressure update equation

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

where $\Delta p = p^{n+1} - \hat{p}$. The operator $\mathbf{E}$ is usually referred to as the consistent pressure Poisson operator. The computational complexity is often dominated by the inversion of this operator in a typical implementation. Hence efficient and reliable inversion is imperative for a fast solver. This is the main focus of our work.

4 Tensor-product forms of the discrete operators

We now consider the tensor-product structure of the discrete operators. We initially identify our unknowns using a global numbering scheme, while the basis functions are enumerated in a mixed local-global scheme, where we use a single, global number within each x_1x_2-plane and a separate number in the x_3-direction.

In the following, let $\phi_m(x_1, x_2)\,\psi_n(x_3)$ be a basis function for one of the velocity components and $\phi_j(x_1, x_2)\,\psi_k(x_3)$ an associated test function. Likewise, let $\tilde{\phi}_m(x_1, x_2)\,\tilde{\psi}_n(x_3)$ be a basis function for the pressure, and
$\tilde{\phi}_j(x_1, x_2)\,\tilde{\psi}_k(x_3)$ be an associated test function.

The elements in the mass matrix are given as

$$\begin{aligned}(\phi_m(x_1,x_2)\psi_n(x_3), \phi_j(x_1,x_2)\psi_k(x_3)) &= \int_\Omega \phi_m\psi_n\phi_j\psi_k \,\mathrm{d}\Omega\\ &= \underbrace{\left(\int_{\mathcal{O}} \phi_m\phi_j \,\mathrm{d}x_1\mathrm{d}x_2\right)}_{B^{2D}_{jm}}\underbrace{\left(\int_0^d \psi_n\psi_k \,\mathrm{d}x_3\right)}_{B^{1D}_{kn}}.\end{aligned}$$

The mass matrix can thus be written as a tensor-product between a 2D and a 1D operator, i.e., the global mass matrix can be expressed as

$$\mathbf{B} = \left(\mathbf{B}^{1D} \otimes \mathbf{B}^{2D}\right).$$

This also means that, according to the rules of tensor-products, we have

$$\mathbf{B}^{-1} = \left(\left(\mathbf{B}^{1D}\right)^{-1} \otimes \left(\mathbf{B}^{2D}\right)^{-1}\right).$$

Similarly, the (scalar) Laplace operator can be expressed as

$$(\nabla\phi_m(x_1,x_2)\psi_n(x_3),\ \nabla\phi_j(x_1,x_2)\psi_k(x_3)) =$$
$$\underbrace{\left(\int_{\mathcal{O}}\tilde{\nabla}\phi_m\cdot\tilde{\nabla}\phi_j\,\mathrm{d}x_1\mathrm{d}x_2\right)}_{A^{2D}_{jm}}\underbrace{\left(\int_0^d\psi_n\psi_k\,\mathrm{d}x_3\right)}_{B^{1D}_{kn}}+\underbrace{\left(\int_{\mathcal{O}}\phi_m\phi_j\,\mathrm{d}x_1\mathrm{d}x_2\right)}_{B^{2D}_{jm}}\underbrace{\left(\int_0^d\psi_n'\psi_k'\,\mathrm{d}x_3\right)}_{A^{1D}_{kn}},$$

where $\tilde{\nabla}$ denotes the gradient operator in the x_1x_2-plane. Again, the resulting global operator can be expressed on tensor-product form as

$$\mathbf{A}=\mathbf{B}^{1D}\otimes\mathbf{A}^{2D}+\mathbf{A}^{1D}\otimes\mathbf{B}^{2D}.$$

Next, we consider the divergence operator. This operator is the sum of three contributions. We consider each contribution separately. For the x_1-component (i.e., differentiation with respect to the x_1-direction) we have

$$\left(\frac{\partial}{\partial x_1}\phi_m\psi_n,\tilde{\phi}_j\tilde{\psi}_k\right)=\underbrace{\left(\int_{\mathcal{O}}\tilde{\phi}_j\frac{\partial}{\partial x_1}\phi_m\,\mathrm{d}x_1\mathrm{d}x_2\right)}_{\left(D^{2D}_1\right)_{jm}}\underbrace{\left(\int_0^d\psi_n\tilde{\psi}_k\,\mathrm{d}x_3\right)}_{\left(B^{1D}_{up}\right)_{kn}},$$

giving rise to the global operator

$$\mathbf{D}_1=\mathbf{B}^{1D}_{up}\otimes\mathbf{D}^{2D}_1.$$

Similarly, the x_2-component yields

$$\mathbf{D}_2=\mathbf{B}^{1D}_{up}\otimes\mathbf{D}^{2D}_2.$$

Finally, for the x_3-component we get

$$\left(\frac{\partial}{\partial x_3}\phi_m\psi_n,\tilde{\phi}_j\tilde{\psi}_k\right)=\underbrace{\left(\int_{\mathcal{O}}\tilde{\phi}_j\phi_m\,\mathrm{d}x_1\mathrm{d}x_2\right)}_{\left(B^{2D}_{up}\right)_{jm}}\underbrace{\left(\int_0^d\tilde{\psi}_k\frac{\partial}{\partial x_3}\psi_n\,\mathrm{d}x_3\right)}_{\left(D^{1D}\right)_{kn}}$$

which assembles to the global operator

$$\mathbf{D}_3=\mathbf{D}^{1D}\otimes\mathbf{B}^{2D}_{up}.$$

Our divergence operator is then given as

$$\mathbf{D}=\begin{bmatrix}\mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3\end{bmatrix},$$

while the gradient operator is given as the transpose.

Inserting the tensor-product form of our operators into the consistent pressure Poisson operator, $\mathbf{E}$, and applying the rules of tensor-products, we get

$$\begin{aligned}\mathbf{E}&=\begin{bmatrix}\mathbf{D}_1 & \mathbf{D}_2 & \mathbf{D}_3\end{bmatrix}\begin{bmatrix}\mathbf{B}^{-1} & 0 & 0\\ 0 & \mathbf{B}^{-1} & 0\\ 0 & 0 & \mathbf{B}^{-1}\end{bmatrix}\begin{bmatrix}\mathbf{D}_1^T\\ \mathbf{D}_2^T\\ \mathbf{D}_3^T\end{bmatrix}\\ &=\mathbf{B}^{1D}_*\otimes\mathbf{E}^{2D}+\mathbf{E}^{1D}\otimes\mathbf{B}^{2D}_*\end{aligned}\tag{2}$$

where

$$
\begin{aligned}
\mathbf{B}_*^{1D} &= \mathbf{B}_{up}^{1D} \left(\mathbf{B}^{1D}\right)^{-1} \left(\mathbf{B}_{up}^{1D}\right)^T, \\
\mathbf{E}^{1D} &= \mathbf{D}^{1D} \left(\mathbf{B}^{1D}\right)^{-1} \left(\mathbf{D}^{1D}\right)^T, \\
\mathbf{B}_*^{2D} &= \begin{bmatrix}\mathbf{B}_{up}^{2D} & \mathbf{B}_{up}^{2D}\end{bmatrix} \begin{bmatrix}\mathbf{B}^{-1} & 0 \\ 0 & \mathbf{B}^{-1}\end{bmatrix} \begin{bmatrix}\left(\mathbf{B}_{up}^{2D}\right)^T \\ \left(\mathbf{B}_{up}^{2D}\right)^T\end{bmatrix} \\
\mathbf{E}^{2D} &= \begin{bmatrix}\mathbf{D}_1 & \mathbf{D}_2\end{bmatrix} \begin{bmatrix}\mathbf{B}^{-1} & 0 \\ 0 & \mathbf{B}^{-1}\end{bmatrix} \begin{bmatrix}\mathbf{D}_1^T \\ \mathbf{D}_2^T\end{bmatrix}
\end{aligned}
\tag{3}
$$

Thus the 3D consistent pressure Poisson operator, $\mathbf{E}$, can also be stated on tensor-product form.

In the following, we will refer to the operators $\mathbf{B}_*^{1D}$ and $\mathbf{B}_*^{2D}$ as "consistent pressure mass matrices".

5 Tensor-product solvers

This work is based on earlier work reported in [1]. We here briefly recall the key steps for the Helmholtz solver.

From the previous section it readily follows that the Helmholtz operator can be expressed as

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

With the boundary conditions considered here, the $\mathbf{A}^{1D}$ and $\mathbf{B}^{1D}$ operators are symmetric and positive definite. We then consider the generalized symmetric eigenvalue problem

$$\mathbf{A}^{1D}\mathbf{Q} = \mathbf{B}^{1D}\mathbf{Q}\mathbf{\Lambda},$$

where the eigenvectors are scaled such that

$$
\begin{aligned}
\mathbf{Q}^T\mathbf{B}^{1D}\mathbf{Q} = \mathbf{I} \;&\Rightarrow \mathbf{B}^{1D} = \mathbf{Q}^{-T}\mathbf{Q}^{-1}, \\
\mathbf{Q}^T\mathbf{A}^{1D}\mathbf{Q} = \mathbf{\Lambda} &\Rightarrow \mathbf{A}^{1D} = \mathbf{Q}^{-T}\mathbf{\Lambda}\mathbf{Q}^{-1}.
\end{aligned}
$$

This is exactly the same eigenvalue problem we would consider if we were constructing a classical fast diagonalization solver in fully tensor-product geometries [17]. Substituting the expressions for $\mathbf{A}^{1D}$ and $\mathbf{B}^{1D}$ we get

$$
\begin{aligned}
\mathbf{H} &= \mathbf{Q}^{-T}\mathbf{Q}^{-1} \otimes \mathbf{A}^{2D} + \mathbf{Q}^{-T}\mathbf{\Lambda}\mathbf{Q}^{-1} \otimes \mathbf{B}^{2D} + \alpha \mathbf{Q}^{-T}\mathbf{Q}^{-1} \otimes \mathbf{B}^{2D} \\
&= \left(\mathbf{Q}^{-T} \otimes \mathbf{I}^{2D}\right) \left(\mathbf{I}^{1D} \otimes \mathbf{A}^{2D} + \left(\mathbf{\Lambda} + \alpha \mathbf{I}^{1D}\right) \otimes \mathbf{B}^{2D}\right) \left(\mathbf{Q}^{-1} \otimes \mathbf{I}^{2D}\right).
\end{aligned}
$$

We now change to a mixed local-global numbering system, where we identify each unknown using a global number in the x_1x_2-plane, while we use a separate number in the x_3-direction. We can then solve the linear system

$$\mathbf{H}u = g,$$

as given in Algorithm 1. Note that the number of subproblems we have to solve depends on the boundary conditions on the top and the bottom of the domain. For example, for homogenous Dirichlet boundary conditions and a single layer of elements, $\mathcal{N}_1 = N - 1$. We want to emphasize one of the main attractive features of the algorithm: the individual subproblems in step 2) of the algorithm are completely decoupled.

We proceed in much the same way to construct a solver for the consistent pressure Poisson operator. With the given Dirichlet boundary conditions for the velocity, the $\mathbf{E}$ operator

is symmetric and positive semi-definite. In particular this applies to the $\mathbf{E}^{1D}$ operator. We have exactly one zero eigenvalue, which is associated with the hydrostatic pressure mode, while all the other eigenvalues are strictly positive. We consider the generalized symmetric eigenvalue problem

$$\mathbf{E}^{1D}\mathbf{Q} = \mathbf{B}_*^{1D}\mathbf{Q}\mathbf{\Lambda},$$

and scale the eigenvectors such that

$$\begin{aligned}\mathbf{Q}^T\mathbf{B}_*^{1D}\mathbf{Q} = \mathbf{I} \Rightarrow \\ \mathbf{Q}^{-T}\mathbf{Q}^{-1} = \mathbf{B}_*^{1D}, \quad \mathbf{E}^{1D} = \mathbf{Q}^{-T}\mathbf{\Lambda}\mathbf{Q}^{-1}.\end{aligned}$$

Inserting this into (2) yields

$$\mathbf{E} = \left(\mathbf{Q}^{-T} \otimes \mathbf{I}^{2D}\right)\left(\mathbf{I}^{1D} \otimes \mathbf{E}^{2D} + \mathbf{\Lambda} \otimes \mathbf{B}_*^{2D}\right)\left(\mathbf{Q}^{-1} \otimes \mathbf{I}^{2D}\right).$$

Switching to our mixed local-global numbering scheme again, we can then solve the problem

$$\mathbf{E}\Delta p = g,$$

as given in Algorithm 2. Again, each of the systems in Step 2 are completely decoupled. The number of systems we have to solve is independent of the boundary conditions for the velocity. For example, for a single layer of elements, $\mathcal{N}_2 = N - 1$.

Note that, except for the case when $\lambda_j = 0$, the two-dimensional pressure operator in Step 2,

$$\mathbf{E}_\lambda^{2D} \equiv \mathbf{E}^{2D} + \lambda_j \mathbf{B}_*^{2D}, \tag{4}$$

is different from the standard two-dimensional pressure operator $\mathbf{E}^{2D}$.

Algorithm 1 Fast tensor-product solver for the Helmholtz system

$$\left(\mathbf{A} + \alpha\mathbf{B}\right) u = g$$

in partially deformed three-dimensional geometries.

Initialization Solve the generalized eigenvalue problem of dimension $\mathcal{N}_1$

$$\mathbf{A}^{1D}\mathbf{Q} = \mathbf{B}^{1D}\mathbf{Q}\mathbf{\Lambda}$$

scaled such that

$$\begin{aligned}\mathbf{B}^{1D} &= \mathbf{Q}^{-T}\mathbf{Q}^{-1} \\ \mathbf{A}^{1D} &= \mathbf{Q}^{-T}\mathbf{\Lambda}\mathbf{Q}^{-1}.\end{aligned}$$

Then

1) Compute
$$\tilde{g} = \mathbf{Q}^T g.$$

2) Solve
$$\left(\mathbf{A}^{2D} + (\lambda_j + \alpha)\mathbf{B}^{2D}\right)\tilde{u}_j = \tilde{g}_j, \qquad \forall\ j = 1, \cdots, \mathcal{N}_1.$$

3) Finally compute the solution as
$$u = \tilde{u}\mathbf{Q}.$$

Algorithm 2 Fast tensor-product solver for the pressure system

$$\mathbf{E}\Delta p = g$$

in partially deformed three-dimensional geometries.

Initialization Solve the generalized eigenvalue problem of dimension $\mathcal{N}_2$

$$\mathbf{E}^{1D}\mathbf{Q} = \mathbf{B}_*^{1D}\mathbf{Q}\mathbf{\Lambda}$$

scaled such that

$$\begin{aligned}\mathbf{B}_*^{1D} &= \mathbf{Q}^{-T}\mathbf{Q}^{-1}\\ \mathbf{E}^{1D} &= \mathbf{Q}^{-T}\mathbf{\Lambda}\mathbf{Q}^{-1}.\end{aligned}$$

Then

1) Compute
$$\tilde{g} = \mathbf{Q}^T g.$$

2) Solve
$$\left(\mathbf{E}^{2D} + \lambda_j \mathbf{B}_*^{2D}\right)\tilde{p}_j = \tilde{g}_j, \qquad \forall\ j = 1, \cdots, \mathcal{N}_2.$$

3) Finally compute the solution as
$$p = \tilde{p}\mathbf{Q}.$$

Extension to other boundary conditions

The pressure operators (3) reflect the velocity boundary conditions. Let us now instead consider the following boundary conditions on the top and the bottom of some domain $\Omega = \mathcal{O} \times (0, 1)$,

$$\begin{aligned}u_1(x_1, x_2, 0) = u_2(x_1, x_2, 0) = u_3(x_1, x_2, 0) &= 0,\\ u_3(x_1, x_2, 1) &= 0,\\ \frac{\partial u_1}{\partial x_3}(x_1, x_2, 1) = \frac{\partial u_2}{\partial x_3}(x_1, x_2, 1) &= 0.\end{aligned} \tag{5}$$

Note that we have not given the boundary conditions on the side walls, since they are of no consequence for the following. The new boundary conditions (5) lead to a different basis being used for the first two velocity components, as opposed to the third component. Carefully considering the integrals given earlier, we again end up with an operator of the form

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

However, this time around the operators $\mathbf{B}_\diamond^{1D}$ and $\mathbf{E}^{1D}$ are constructed from different bases. In particular, $\mathbf{E}^{1D}$ involves the basis for the third velocity component, while $\mathbf{B}_\diamond^{1D}$ involves the basis for the first two components. That is, $\mathbf{B}_\diamond^{1D}$ is constructed from a velocity mass matrix which does not include a homogenous boundary condition on the top of the domain. Thus we see that none of this influences the actual algorithm, only the formation of the one-dimensional operators. In particular, if we keep the Dirichlet boundary conditions on the side walls, the 2D operators defined earlier are left unchanged.

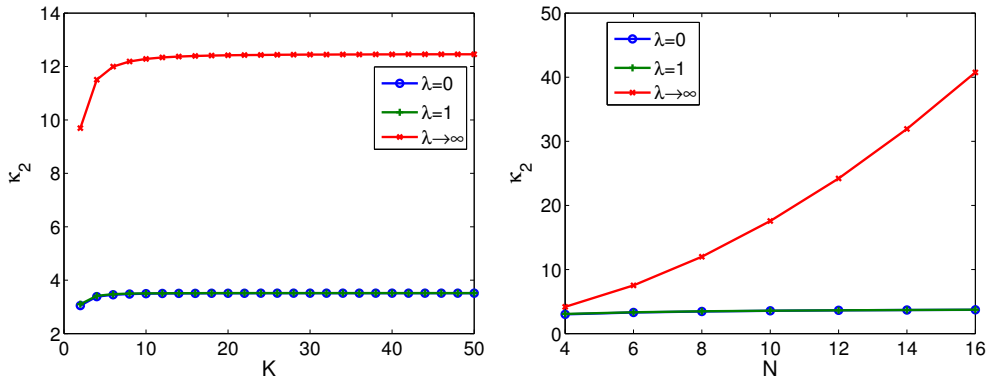
The discussion here also reveals a restriction in the applicability of the method. The boundary conditions for the first two velocity components need to be of the same type (Dirichlet or Neumann), both along the top surface and along the bottom surface. If this is not the case, we cannot state the operator on tensor-product form.

Extension to higher temporal order

In principle, the computational approach readily extends to higher order BDF schemes. However, the use of the splitting scheme brings some restrictions. The pressure correction schemes are only unconditionally stable for first and second order realizations. While a third order realization is conditionally stable under a $\Delta t \gtrsim \mathcal{O}(N^{-4})$ limitation on the time step [11], we do not consider it here. The main reason is that this limitation can lead to unconditionally unstable schemes when combined with a semi-explicit method for integrating the Navier-Stokes equations. The explicit treatment of the convection operator imposes a CFL restriction on the time step. We may then end up in a situation where there is no overlap between the time step restriction imposed by the CFL condition and the time step limitation required by the splitting scheme.

6 A preconditioner for the 2D systems

In the previous section we showed how to solve the full 3D elliptic systems through inversion of several independent 2D systems. If the individual 2D systems are sufficiently small, they can be solved using a direct method. This would mean that the proposed algorithm can be classified as a direct solution method. In many applications though, these systems will be too large to be solved using direct methods. In this case we have to resort to an iterative solution strategy for the individual subproblems. As previously discussed, the systems we consider are either symmetric positive definite or symmetric positive semi-definite. Hence, the conjugate gradient method can be used in combination with a suitable preconditioner.



(a) $N = 8$. (b) $K = 5$.

Figure 3: The condition number of the one-dimensional operator $\mathbf{R}^T(\mathbf{H}^{1D})^{-1}\mathbf{R}\mathbf{E}_\lambda^{1D}$. In (a) the condition number is plotted as a function of the number of spectral elements, K, and in (b) as a function of the element order, N. Here $\mathbf{R}^T$ is the prolongation operator which interpolates from a GL to a GLL point distribution within each element, and $\mathbf{R}$ the corresponding restriction operator. Three different values of λ are considered.

Earlier studies have shown that the $\mathbf{E}$ operator is close to the $\mathbf{A}$ operator (in the sense that their spectra resemble each other), e.g., see [7]. Inspired by this, we expect the $\mathbf{E}_\lambda^{2D}$ operator in (4) to be close to the two-dimensional Helmholtz operator, $\mathbf{H}^{2D} = \mathbf{A}^{2D} + \lambda_j \mathbf{B}^{2D}$. To test this idea, we first consider a simple one-dimensional test where we use $\mathbf{H}^{1D} = \mathbf{A}^{1D} + \lambda_j \mathbf{B}^{1D}$ as a preconditioner for $\mathbf{E}_\lambda^{1D} = \mathbf{E}^{1D} + \lambda_j \mathbf{B}_*^{1D}$. This seems to work well for moderate values of λ; see Figure 3. However, the growth with the order of the elements, N, for large λ is a source of concern. This growth is not entirely unexpected either since our preconditioner enforces continuity across the element boundaries, while the pressure is discontinuous. Nonetheless, these results are quite encouraging since they show that we achieve good preconditioning for moderate values of λ, at least in one space dimension.

To improve the preconditioning for all values of λ, our preconditioner will be composed of two different preconditioners, one used for small to moderate values of λ and one used for large values of λ. The first part of the preconditioner is based on the work presented in [7] and represents a preconditioner based on the (additive) overlapping Schwarz methodology [6, 26]. We introduce a coarse grid with K_C subdomains, independent of the number of degrees of freedom on the fine grid. Each subdomain can span one or several spectral elements on the fine grid. The preconditioner is then constructed as a sum of local, finite-element based solves within the subdomains and a global, coarse solve on the fine grid. This preconditioner can be expressed as

$$\mathbf{M}_1^{-1} = \sum_{k=1}^{K_C} \mathbf{R}_{k,C}^T \tilde{\mathbf{H}}_k^{-1} \mathbf{R}_{k,C} + \mathbf{R}_{0,C}^T \mathbf{H}_0^{-1} \mathbf{R}_{0,C},$$

where the local operators $\tilde{\mathbf{H}}_k$ are based on (tensorized) linear finite element discretizations on the pressure grid and the coarse operator $\mathbf{H}_0$ is based on quadratic elements. In our case, the local solves are performed using fast (two-dimensional) tensor-product solvers even though the subdomains will not (in general) be rectangles. The global solve is performed using Cholesky factorization. The coarse restriction operator $\mathbf{R}_{0,C}$ is the composition of two restriction operators $\mathbf{R}_{0,A}$ and $\mathbf{R}_{0,B}$, where $\mathbf{R}_{0,A}$ is taken such that the prolongation operator $\mathbf{R}_{0,A}^T$ is the interpolation operator which does interpolation from second order polynomials to Nth order polynomials within each element, and $\mathbf{R}_{0,B}$ is taken such that the prolongation operator $\mathbf{R}_{0,B}^T$ is the interpolation operator between the velocity and pressure grid within each element. The local restriction operator $\mathbf{R}_{k,C}$ restricts the input (the residual) to the specific subdomain considered ($k = 1, \cdots, K_C$).

From the results in Figure 3, we expect the efficiency of this preconditioner to degrade as λ grows. The second part of our preconditioner is designed to deal with this deficiency. For large values of λ, the $\mathbf{E}_\lambda$ operator approaches the consistent pressure mass matrix, $\mathbf{B}_*$ (up to a multiplicative factor). Inspired by the preconditioning of the Uzawa pressure operator in the steady Stokes context [18], we consider a preconditioning strategy for large values of λ based on the standard (diagonal) pressure mass matrix, $\tilde{\mathbf{B}}$,

$$\mathbf{M}_2^{-1} = \tilde{\mathbf{B}}^{-1}.$$

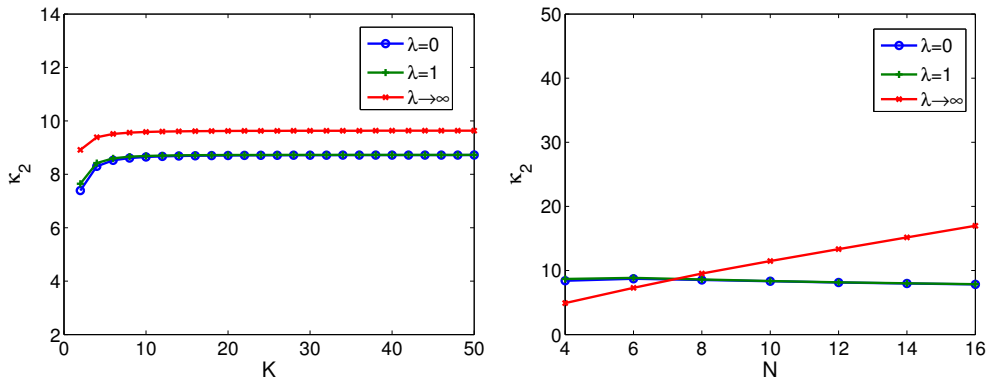
We have also tried several other preconditioning strategies, including some of the variants proposed in [7], as well as the deflation-based preconditioner proposed in [23] (only suitable for $\lambda = 0$). Our experiences are consistent with those reported earlier in the literature and we will not discuss these alternatives further here. A final class of preconditioners is the multi-grid-based methods [16]; we have not tried any preconditioner from this class.

7 Numerical results

We start with analyzing the two preconditioners under idealized conditions. We then give convergence results and speedup results in more realistic settings. Finally, we solve the three-dimensional Bénard-Marangoni problem in a hexagonal container using the proposed solver. In Section 8, we report some results where we have used the tensor-product solver as a preconditioner in nonextruded domains.

Preconditioning

We first consider preconditioning the $\mathbf{E}_\lambda$ operator in 1D using the $\mathbf{M}_1$ preconditioner. In these idealized tests the number of subdomains used in the preconditioner and the number of spectral elements are taken to be equal (i.e., $K_C = K$). The results are presented in Figure 4.



(a) $N = 8$. (b) $K = 5$.

Figure 4: The condition number of the one-dimensional operator $\mathbf{M}_1^{-1}\mathbf{E}_\lambda^{1D}$. In (a) the condition number is plotted as a function of the number of spectral elements, K, and in (b) as a function of the element order, N. Three different values of λ are considered.

These results are somewhat surprising. Comparing Figures 3 and 4, it seems that $\mathbf{M}_1$ is a slightly better preconditioner for $\mathbf{E}_\lambda^{1D}$ compared to the global Helmholtz operator $\mathbf{H}^{1D}$. This can probably be explained by the fact that we do the local solves directly on the pressure grid rather than on the velocity grid. Qualitatively we see that we have the same behavior as when using the full Helmholtz operator; the condition number is uniformly bounded as a function of K, but has growth as a function of N for large λ. These one-dimensional results also translate to two space dimensions as shown in Figure 5. In these tests, the two-dimensional domain is a square comprised of $K = K_1 \times K_1$ spectral elements of order N.

The preconditioner $\mathbf{M}_1$ is suboptimal for large λ. For large λ, the operator $\mathbf{E}_\lambda^{2D}$ will approach $\mathbf{B}_*^{2D}$ (up to a multiplicative constant). This operator can again be approximated by the (diagonal) pressure mass matrix, $\tilde{\mathbf{B}}$. For λ larger than some threshold value we should switch to the preconditioner $\mathbf{M}_2$, both because of the lower condition number and because of the lower computational cost; see Figure 6.

To estimate where the cross-over value of λ is, we can consider a scaling argument. Numerical results indicate that the condition number of the consistent pressure Poisson

operator scales as

$$\kappa_2\left(\mathbf{E}^{2D}\right) \sim c_1 K_1^2 N^3,$$

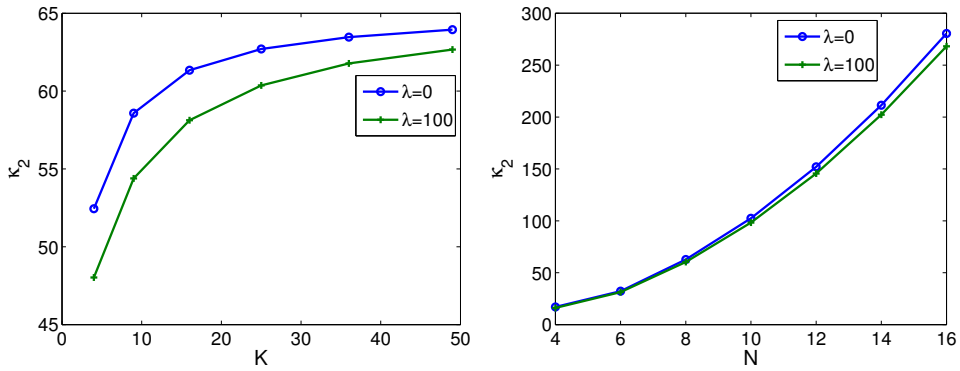
while the condition number of the consistent pressure mass matrix scales as

$$\kappa_2\left(\mathbf{B}_*^{2D}\right) \sim c_2 N^3.$$

Balancing these two terms gives

$$c_1 K_1^2 N_3 = \lambda c_2 N^3 \Rightarrow \lambda = \frac{c_1}{c_2} K_1^2.$$

Hence, the cross-over value of λ should be independent of N, and only depend on K_1, the number of elements in one spatial direction in the two-dimensional plane. Numerical tests confirm these predictions.



(a) $N = 8$. (b) $K = 7 \times 7 = 49$.

Figure 5: The condition number of the two-dimensional operator $\mathbf{M}_1^{-1}\mathbf{E}_\lambda^{2D}$. In (a) the condition number is plotted as a function of the number of spectral elements, K, and in (b) as a function of the element order, N. Two different values of λ are considered.

Convergence results in a circular cylinder

We now perform a convergence test to confirm the expected exponential convergence in space as well as the expected temporal order.

The geometry we consider is a circular cylinder. We divide our geometry into $K = 48$ spectral elements, with a single layer of elements in the x_3-direction, see Figure 7(a). The preconditioner is based on a $K_C = 32$ subdomain division of the computational grid; see Figure 7(b). As initial conditions at $t = 0$ we take those that are consistent with the exact unsteady Stokes solution (here in cylindrical coordinates)

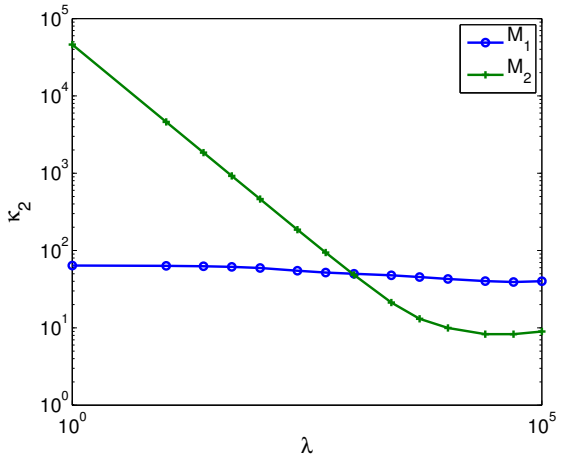


Figure 6: The condition number of the two-dimensional operator $\mathbf{M}_i^{-1}\mathbf{E}_\lambda^{2D}$, $i = 1, 2$, as a function of λ. Here, $K = 49$ and $N = 8$. For $\lambda > 10^3$, $\mathbf{M}_2^{-1}$ yields the lowest condition number and has the lowest computational cost.

$$
\begin{aligned}
u_r(r,z,\theta,t) &= \frac{1}{5}\sin^2(\pi r)\sin(2\pi z)\sin\theta\sin t \\
u_\theta(r,z,\theta,t) &= \frac{1}{5}\sin(\pi r)\left(2\pi r\cos(\pi r)+\sin(\pi r)\right)\cos\theta\sin(2\pi z)\sin t \\
u_z(r,z,\theta,t) &= \frac{1}{10\pi}\sin(\pi r)\left(2\pi\cos(\pi r)+\frac{2}{r}\sin(\pi r)\right)\sin\theta\left(\cos(2\pi z)-1\right)\sin t \\
p(r,z,\theta,t) &= \sin^2(\pi r)\sin t
\end{aligned}
\tag{6}
$$

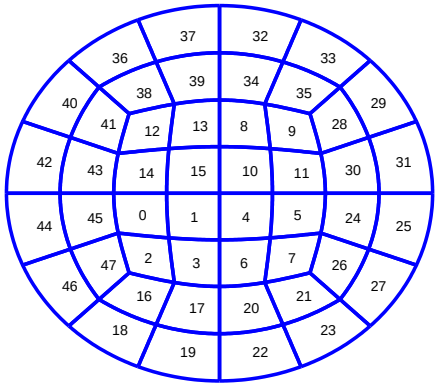
where

$$
r = \sqrt{x^2+y^2}, \qquad \sin\theta = \frac{y}{r}, \qquad \cos\theta = \frac{x}{r}.
$$

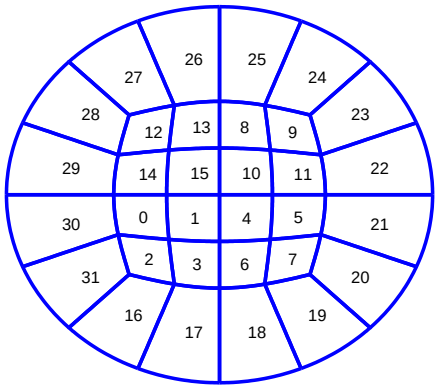
In order to compare our results with the exact solution we use the source function $\mathbf{f}$ that corresponds to this solution.

The pressure-velocity splitting scheme introduce splitting errors which are of the same order as the BDF scheme employed, which means we expect full first and second order convergence for the velocity measured in the L^2 norm. However, while numerical evidence shows that we obtain full second order convergence in the H^1 norm as well, current theory shows that we can only expect $3/2$ order convergence for general problems [10].

For the pressure, however, the story is a bit more complicated. Current theory shows that we can, in general, only expect a convergence rate of $\Delta t^{1/2}$ for a first order realization and a convergence rate of $\Delta t^{3/2}$ in a second order realization if we measure the error in the L^2-norm [11]. Numerous numerical test cases have shown that this estimate is overly conservative and that we typically get the full expected (first/second) order. Here, we only give convergence results for the velocity; see Figure 8. We observe that we obtain the expected order of convergence; first and second order in time, and spectral convergence in space for smooth solution and data.

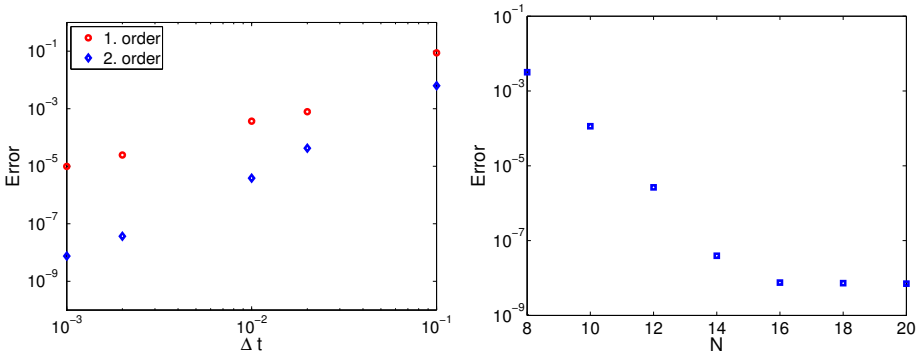


(a) Domain decomposition: spectral elements



(b) Domain decomposition: preconditioner

Figure 7: 2D cross-sections of the cylinder we consider. In (a) we show the decomposition of the computational domain into spectral elements (here $K = 48$), while in (b) we show the subdomains used in the preconditioner $\mathbf{M}_2$ (here $K_C = 32$).



(a) Temporal (b) Spatial

Figure 8: The left plot depicts the discretization error of the velocity measured in the discrete H^1-norm at time $t = 1$ as a function of the time step, Δt, for a first and second order temporal splitting scheme. The right plot depicts the discretization error of the velocity measured in the discrete H^1-norm at time $t = 1$ as a function of the polynomial degree of the elements, N. We observe the expected exponential convergence. The temporal error is here subdominant for $N < 16$.

Efficiency on a single processor

We are now ready to compare the new tensor-product solver with a more conventional iterative solver. Our 3D reference solvers (for the Helmholtz problem and the consistent pressure Poisson problem, respectively) will be based on the additive Schwarz method using minimal overlap; for the consistent pressure Poisson operator we will use the method proposed in [7]. However, note that the 3D reference solver for the pressure involves the 3D operator $\mathbf{E}$, while the decoupled 2D systems in the new tensor-product solver involve the operator $\mathbf{E}_\lambda^{2D}$ (for different values of λ). In the reference solvers, we also take into account the particular geometries we study, i.e., any geometric terms we know are zero are eliminated. The reason for this is to emphasize that any observed speedup is not just caused by doing less work due to having eliminated specific geometric terms in the new solver. Hence, the speedup reported will be conservative since in a general 3D spectral element code this elimination would most likely not be performed.

We consider two simple test problems, one for each of the operators. For the Helmholtz problem $-\nabla^2 u + \alpha u = f$, we use a right hand side

$$f = 1. \tag{7}$$

To test the consistent pressure operator, we consider a test problem with a body force $\mathbf{f} = (f_1, f_2, f_3)^T$, with

$$f_1 = f_2 = 0, \quad f_3 = -1. \tag{8}$$

This body force is compatible with a zero velocity and a pressure which is linear in the x_3-direction in all "extruded" geometries. We do a single time step with $\Delta t = 1$.

The domain is a circular container. The specific shape of this container is determined by the global aspect ratio

$$\Gamma = \frac{\sqrt{A}}{d},$$

where A denotes the top surface area and d the height of the container. In the following

we consider two different cases:

$$\text{Container 1: } \Gamma = 1,$$
$$\text{Container 2: } \Gamma = 5.$$

Table 1: A 3D Helmholtz problem ($\alpha = 100$) is solved in Container 1 using (7) as the source function. In (a) we report the mean number of iterations for solving the two-dimensional subproblems in Algorithm 1, both when using unpreconditioned conjugate gradient iterations (CG) or preconditioned conjugate gradient (PCG) iterations. In (b) we report the number of iterations for the reference 3D solver. In all cases a tolerance of $\|r\| < 10^{-10}$ has been used. The largest grid ($K = 768$, $N = 16$) has about 2.5 million grid points.

(a) Tensor-product solver (Algorithm 1)

	$k = 48$		$k = 192$		$k = 768$	
N	CG	PCG	CG	PCG	CG	PCG
4	79	33	158	45	321	58
6	111	42	221	56	450	76
8	146	51	291	72	595	102
10	178	60	354	87	723	124
12	213	69	420	102	854	146
14	248	77	485	116	979	166
16	282	85	550	129	1101	185

(b) 3D reference solver

	$k = 48$		$k = 192$		$k = 768$	
N	CG	PCG	CG	PCG	CG	PCG
4	47	41	87	58	181	82
6	82	64	154	96	318	140
8	121	91	231	137	479	206
10	168	117	324	183	666	276
12	221	145	428	230	881	348
14	282	174	541	278	1112	421
16	347	204	663	324	1361	491

We first consider the efficiency of our preconditioners in terms of the number of iterations. We start with the Helmholtz problem to confirm that we have the expected behavior. In all these tests we put $\alpha = 100$, which should be close to the value it would have in a Stokes problem, assuming a time step $\Delta t \approx 10^{-2}$. The results for Container 1 are given in Table 1. Since our preconditioners use minimal overlap we see that the number of iterations grows approximately linearly with N. This is as expected since near the edges the Gauss point distribution have a spacing which is $\mathcal{O}(N^2)$. This means that the condition number will grow as $\mathcal{O}(N^2)$ which translates into a $\mathcal{O}(N)$ behavior for the iteration counts.

Results are reported for $K = 48$, $K = 192$ and $K = 768$ spectral elements. Each refinement has been constructed by splitting the two-dimensional cross-section of each spectral element into 4 new elements. Since we use a fixed number of subdomains ($K_C = 32$) in all cases, each refinement means that the overlap between the subdomains will be reduced by a factor of two, and hence the condition number will increase with a factor of

two, which again translates into an increase in the number of iterations with a factor $\sqrt{2}$. This agrees quite well with the numerical results.

Table 2: The consistent pressure Poisson problem is solved in Container 1 using (8) as the body force. A single time step with $\Delta t = 1$ is performed. In (a) we report the number of iterations used when solving the first plane with $\lambda_j > 0$ in Step 2 in Algorithm 2. In (b) we report the number of iterations for the reference 3D solver. In all cases a tolerance of $\|r\| < 10^{-10}$ has been used. The largest grid ($K = 768$, $N = 16$) has about 2.5 million grid points.

(a) Tensor-product solver (Algorithm 2)

	$k = 48$		$k = 192$		$k = 768$	
N	CG	PCG	CG	PCG	CG	PCG
4	126	33	265	45	544	55
6	215	40	445	50	907	64
8	250	43	472	54	940	71
10	296	51	612	67	1245	98
12	378	61	783	82	1598	129
14	464	70	966	102	1983	168
16	556	81	1153	126	2392	206

(b) 3D reference solver

	$k = 48$		$k = 192$		$k = 768$	
N	CG	PCG	CG	PCG	CG	PCG
4	202	127	446	184	935	264
6	393	138	838	212	1717	311
8	525	213	1101	324	2265	468
10	597	261	1246	387	2578	559
12	630	298	1323	440	2723	678
14	773	358	1617	545	3366	843
16	932	423	1942	656	4056	1027

Note that we have here used non-optimal local solvers, both in the context of solving the independent 2D systems in the new solver and in the context of using the 3D reference solver; this derives from the fact that each subdomain is approximated as a rectangle (in the 2D context) or as a hexahedron (in the 3D context). The advantage of this approach is fast, local, tensor-product solvers with minimum memory requirement. The disadvantage is that the iteration count is somewhat higher compared to using local operators which perfectly reflect the geometry of the subdomains. However, our main interest is to construct a fair comparison of the performance of the new solver relative to a suitable reference solver.

We now consider the consistent pressure Poisson operator. The results obtained for Container 1 are presented in Table 2. We observe the same behavior with respect to K and N as we did for the Helmholtz operator. Again, the absolute iteration count could have been reduced a bit with the use of local solvers which perfectly reflect the geometry of the subdomains.

Table 3: A 3D Helmholtz problem ($\alpha = 100$) is solved using (7) as the source function. We present timing results (in seconds) as well as the speedup for the tensor-product solver relative to the 3D reference solver. In (a) we report results for Container 1, while in (b) we report results for Container 2.

(a) Container 1

		$k = 48$		$k = 192$		$k = 768$	
N	solver	CG	PCG	CG	PCG	CG	PCG
8	3D reference	0.30	0.45	2.72	3.01	26.0	20.7
	Tensor-product	0.073	0.094	0.54	0.60	5.87	4.12
	Speedup	4.1	4.79	5.03	5.01	4.43	5.02
16	3D reference	16.5	14.5	163	105	1403	686
	Tensor-product	2.33	1.24	24.9	9.18	225	59.7
	Speedup	7.08	11.7	6.55	11.4	6.23	11.5

(b) Container 2

		$k = 48$		$k = 192$		$k = 768$	
N	solver	CG	PCG	CG	PCG	CG	PCG
8	3D reference	1.00	1.42	8.45	9.65	72.0	62.8
	Tensor-product	0.20	0.28	1.85	1.96	23.0	14.4
	Speedup	5.00	5.07	4.57	4.92	3.13	4.36
16	3D reference	6.38	7.32	52.3	53.6	434	350
	Tensor-product	1.07	1.03	9.62	7.64	76.5	49.7
	Speedup	5.96	7.1	5.44	7.02	5.67	7.04

Next, we present timing and speedup results on a single processor. We solve the same problems as earlier, but now consider the wall clock time. We consider both unpreconditioned operators and preconditioned operators to make sure the speedups obtained are due to the algorithm itself, and not due to technicalities of the preconditioners. The results obtained for the Helmholtz problem are given in Table 3, while Table 4 reports the results for the consistent pressure Poisson problem. Note that the timing results for the new solver represent the total time to solve *all* the two-dimensional problems for each single 3D problem. It is interesting to note that for both problems, the tensor-product solver gives a better speedup in the preconditioned case than in the unpreconditioned case for Container 1. This effect seems to be less pronounced for Container 2. An explanation for this observation will be discussed shortly.

The speedup for the Helmholtz problem is slightly lower than what has been reported earlier. The observed speedup for the new tensor-product solver is approximately 5-7 for the largest aspect ratio considered, while the results in [1] indicate a speedup of around 8-10. This is most likely due to the fact that the reference solver used in [1] involved the full 3D Helmholtz operator (i.e., without explicitly removing the terms we know are zero due to the particular geometries we consider).

Preconditioning effect: 2D versus 3D

The speedup results indicate a sensitivity to the shape of the domain (in this case the global aspect ratio). For example, Table 4 indicate a speedup of about 20 for Container 1 and about 10 for Container 2 (preconditioned case and $N = 8$). We suspect that the

difference is due to the difference in the element/subdomain aspect ratios. In an attempt to illuminate the observed effect, we consider the condition number of the preconditioned pressure Poisson operator in a rectangular geometry. We consider two different grids, and investigate numerically how the aspect ratio of the domain influences the condition numbers. In the first grid, denoted as grid A, the rectangle is divided into a square number of equal-sized elements. In addition to the number of elements, K, and the polynomial degree of the elements, N, we have one more parameter given by

$$\Gamma = \frac{L_1}{L_2},$$

where L_1 and L_2 are the lengths of the domain in the x_1 and x_2-direction, respectively. The parameter Γ gives the global aspect ratio of the domain. The numerical results for grid A for two different aspect ratios are reported in Figure 10. This shows the expected dependence of the condition number on the aspect ratio – higher aspect ratios lead to severe degradation in the preconditioning effect [7]. Note that the local aspect ratios Γ_k, $k = 1, \ldots, K$, are here precisely equal to the global aspect ratio Γ since

$$\Gamma_k = \frac{\frac{L_1}{\sqrt{K}}}{\frac{L_2}{\sqrt{K}}} = \Gamma.$$

The relevance to the "real" grid is that this shows the importance of keeping the subdomain aspect ratios close to unity.

Table 4: The consistent pressure Poisson problem is solved using (8) as the body force. A single time step with $\Delta t = 1$ is performed. We present timing results (in seconds) as well as the speedup for the tensor-product solver relative to the 3D reference solver. In (a) we report results for Container 1, while in (b) we report results for Container 2.

(a) Container 1

		$k = 48$		$k = 192$		$k = 768$	
N	solver	CG	PCG	CG	PCG	CG	PCG
8	3D reference	5.51	5.20	49.9	33.5	434	210
	Tensor-product	0.506	0.243	4.70	1.71	46.7	11.1
	Speedup	10.9	21.4	10.6	19.6	9.30	18.9
16	3D reference	138	114	1165	818	9996	5911
	Tensor-product	9.95	4.63	91.0	35.8	795	277
	Speedup	13.9	24.6	12.8	22.8	12.6	21.3

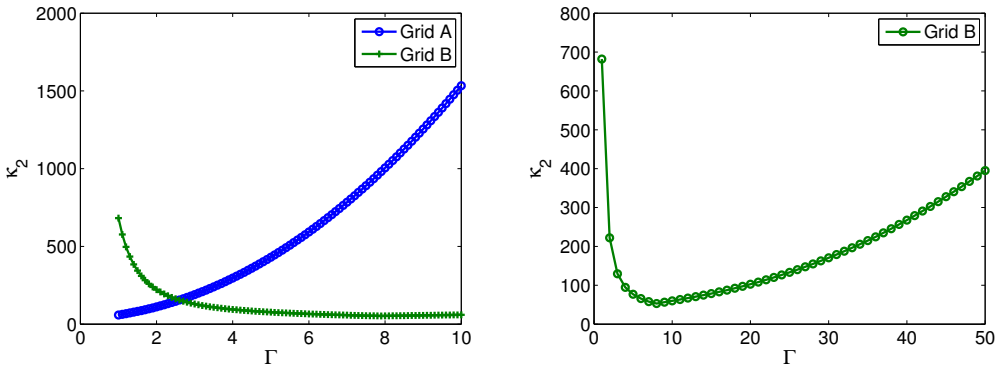
(b) Container 2

		$k = 48$		$k = 192$		$k = 768$	
N	solver	CG	PCG	CG	PCG	CG	PCG
8	3D reference	3.18	2.66	28.4	16.9	244	101
	Tensor-product	0.300	0.232	2.66	1.60	26.1	10.5
	Speedup	10.6	11.5	10.7	10.6	9.35	9.62
16	3D reference	79.8	56.6	663	381	5665	2566
	Tensor-product	5.57	3.42	46.1	24.4	382	168
	Speedup	14.3	16.5	14.3	15.6	14.8	15.3

In the second grid, denoted as grid B, we divide the rectangle into a strip of elements. That is, we have only one layer of elements in the x_2-direction. The effect of the aspect ratio in this case can be seen in Figure 11. We observe that the effects are the opposite here, that is, a high global aspect ratio gives a better preconditioning effect. The reason for this is easy to see if we consider the local aspect ratios,

$$\Gamma_k = \frac{\frac{L_1}{K}}{L_2} = \frac{\Gamma}{K}.$$

Now the local aspect ratios are dependent on both Γ and the number of elements. Hence for a large global aspect ratio Γ, the local aspect ratios improve with an increasing number of elements, until we reach unity, at which point we again have degradation. This is made even clearer in Figure 9 where we plot the condition number in the two geometry configurations as a function of the global aspect ratio $\Gamma = \frac{L_1}{L_2}$ for the consistent pressure Poisson operator.

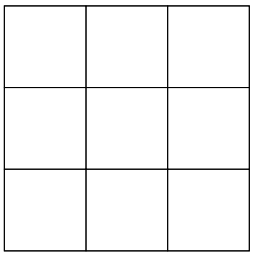


(a) Grid A & Grid B

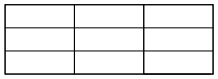
(b) Grid B only

Figure 9: Condition number for the pressure operator using the two different grids as a function of Γ, the global domain aspect ratio. We use $N = 8$ and $K = 9$, while we have left $\lambda = 0$ since this is where the effect is most pronounced. In (a) we observe that the aspect ratio have the opposite effect on the two mesh configurations. In (b) we show an extended plot for the strip geometry to indicate the growth for large Γ.

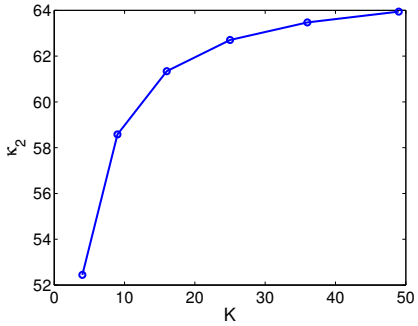
This explains why we observe the diminishing gain from using the tensor-product solver for the larger aspect ratio containers. The topology of the elements in the x_3-direction is exactly as in the strip case; by increasing the global aspect ratio, we get local aspect ratios in the elements which are closer to unity. This property is something only the 3D reference solver benefits from. The tensor-product solver only "sees" the 2D grid where we have no such benefit; the local aspect ratio of the elements are the same no matter what the global aspect ratio is.



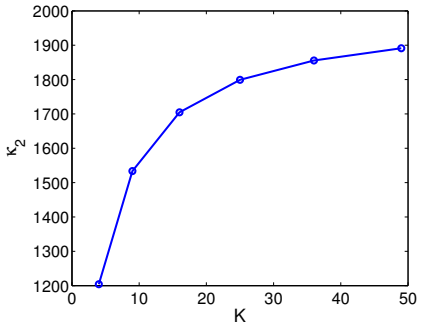
(a) $\frac{L_1}{L_2} = 1$



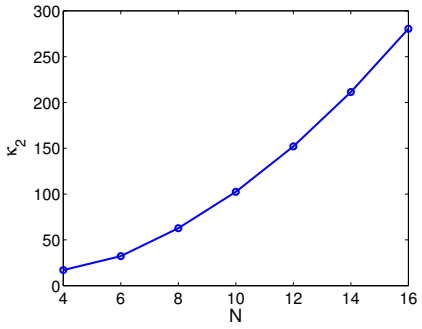
(b) $\frac{L_1}{L_2} = 10$



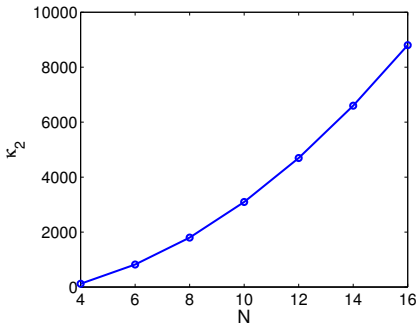
(c) κ_2 as a function of K.



(d) κ_2 as a function of K.

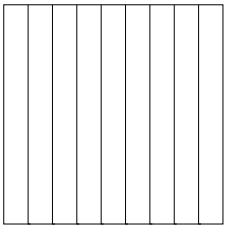


(e) κ_2 as a function of N.

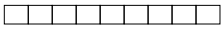


(f) κ_2 as a function of N.

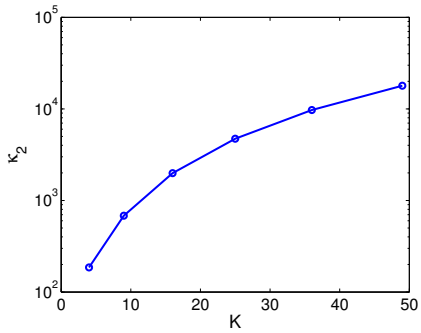
Figure 10: Condition numbers κ_2 for the preconditioned consistent pressure operator $\mathbf{E}^{2D}$ using grid A. The left column shows the results obtained with a global aspect ratio $\Gamma = 1$, while the right column shows the results obtained with a global aspect ratio $\Gamma = 10$. We observe that a higher global aspect ratio results in a higher condition number. In (c) and (d) we have kept N fixed at $N = 8$, while in (e) and (f) we have kept K fixed at $K = 3 \times 3 = 9$.



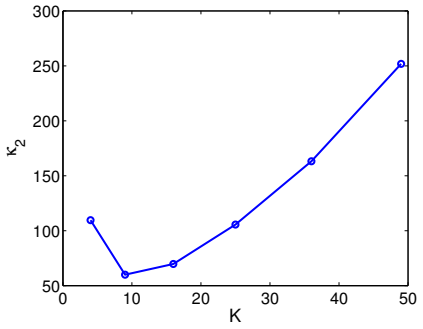
(a) $\frac{L_1}{L_2} = 1$



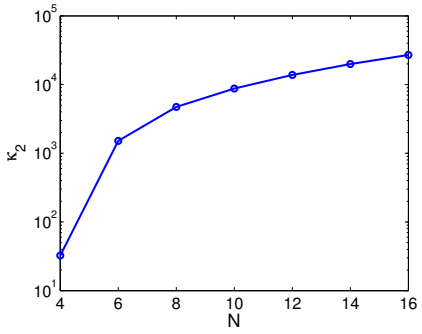
(b) $\frac{L_1}{L_2} = 10$



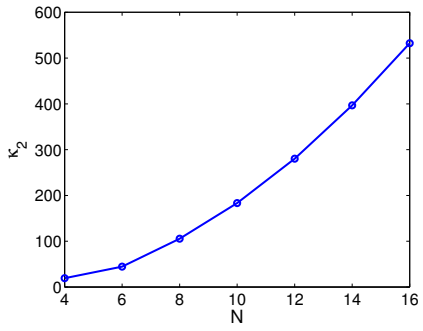
(c) κ_2 as a function of K.



(d) κ_2 as a function of K.



(e) κ_2 as a function of N.



(f) κ_2 as a function of N.

Figure 11: Condition numbers κ_2 for the preconditioned consistent pressure operator $\mathbf{E}^{2D}$ using grid B. The left column shows the results obtained with a global aspect ratio $\Gamma = 1$, while the right column shows the results obtained with a global aspect ratio $\Gamma = 10$. We observe that a higher global aspect ratio results in a lower condition number. In (c) and (d) we have kept N fixed at $N = 8$, while in (e) and (f) we have kept K fixed at $K = 3 \times 3 = 9$.

Bénard-Marangoni simulation

Finally we show some results where the new tensor-product algorithm has been used to solve a real problem. Specifically, we consider three-dimensional Bénard-Marangoni convection in a hexagonal container [14, 21]. This is a coupled fluid-thermal problem with a rich set of solutions depending on the pertinent nondimensional numbers. Figure 12 shows the temperature on the top of the domain after long time integration. The flow organizes itself in such a way that we get hexagonal convection cells with hot fluid at the center of each cell, and colder fluid along the borders. The result were obtained using a parallel implementation of the algorithms [15].

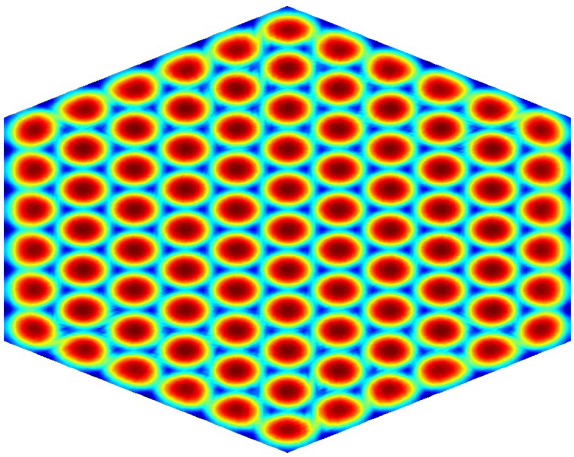


Figure 12: The steady state pattern for a particular Bénard-Marangoni simulation; see [21] for a detailed discussion of this type of problem. The plot shows the temperature along the top of the domain. We observe the characteristic hexagonal cells. The nondimensional numbers used here are Marangoni number $Ma = 105$, Rayleigh number $Ra = 48$, Prandtl number $Pr = 890$, and a global aspect ratio $\Gamma = 30$.

8 The new tensor-product solver as a preconditioner

We now consider computational domains where the assumption about an "extruded" geometry is only approximately satisfied. In such cases, the proposed solver may be used as a global preconditioner in a conjugate gradient iteration. This idea is quite natural and inspired by the fact that fast tensor-product solvers have been much used as building blocks (local subdomain solvers) in Schwarz-type preconditioners [27, 7].

We consider two kinds of deformations. The first is a tapered cylinder, where the relevant parameter is the taper ratio

$$\gamma = \frac{d}{r_1 - r_2};$$

see Figure 13. By this definition the taper ratio approaches infinity as we approach a straight cylinder. This kind of geometry have received some attention in the CFD community lately, in particular in studies of flow around marine risers [22].

We again use the test problems described earlier. The efficiency is now measured as the speedup compared to a standard 3D implementation. We test the preconditioning effect

on two different grid configurations. The first configuration consists of $K = 48$ spectral elements where each element is of order $N = 16$. The 3D reference solver is based on the additive Schwarz algorithm described earlier [7]. The numerical results for this geometry are given in Figure 15(a) and Figure 15(b). The results are quite encouraging, in particular for the consistent Poisson operator, with speedup close to 5 for relatively small taper ratios.

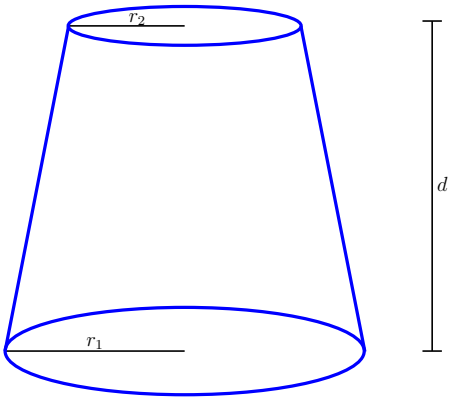
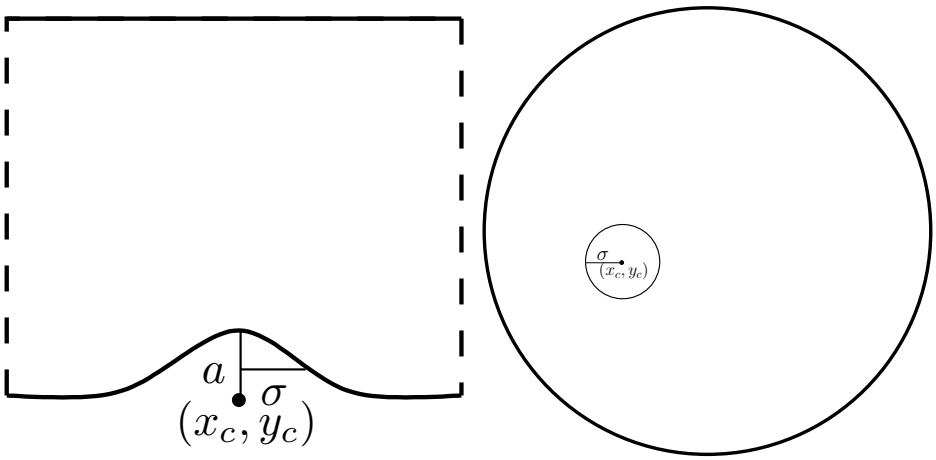


Figure 13: Illustration of a tapered cylinder. The taper ratio is defined as the ratio between the cylinder height and the difference in the radii. The smaller the ratio, the larger the deviation from a straight cylinder.

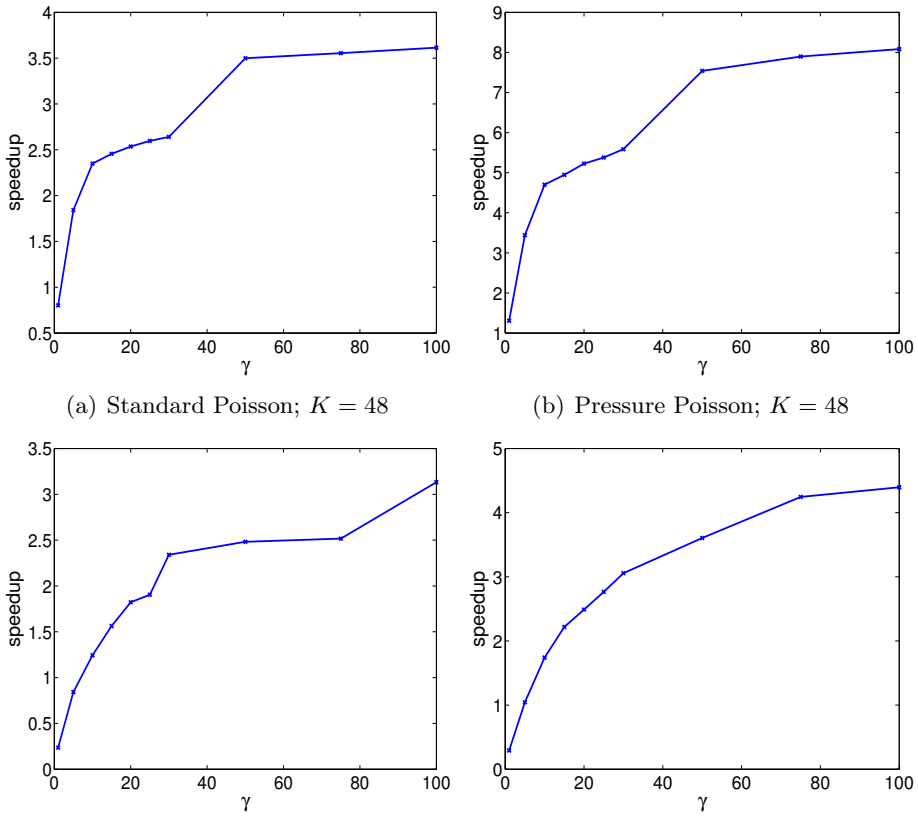


(a) Side view (b) Top view

Figure 14: a) Side view of a cylinder with a bump at the bottom. The bump is a Gaussian centered in (x_c, y_c) with standard deviance σ. The parameter we study is the amplitude a. b) Top view of a cylinder with a bump at the bottom, showing a sample placement of the bump.

The second configuration consists of $K = 480$ spectral elements organized in 10 layers, with each element of order $N = 4$. The results for this geometry are given in Figure 15(c) and 15(d). For the Laplacian operator it seems that the preconditioning effect is fairly invariant with respect to the discretization grid, i.e., about the same for a few high order elements as for many elements of low order. While the consistent pressure Poisson operator

seems to benefit more from using high order elements, the results in the low order context are still quite good. We remark that the reason for the plateaus in the speedup curves for the high order elements, is that very few iterations are performed. For each plateau the iteration count has been reduced by one. In the low order context, this effect is less pronounced, hence we get smoother speedup curves.



(a) Standard Poisson; $K = 48$

(b) Pressure Poisson; $K = 48$

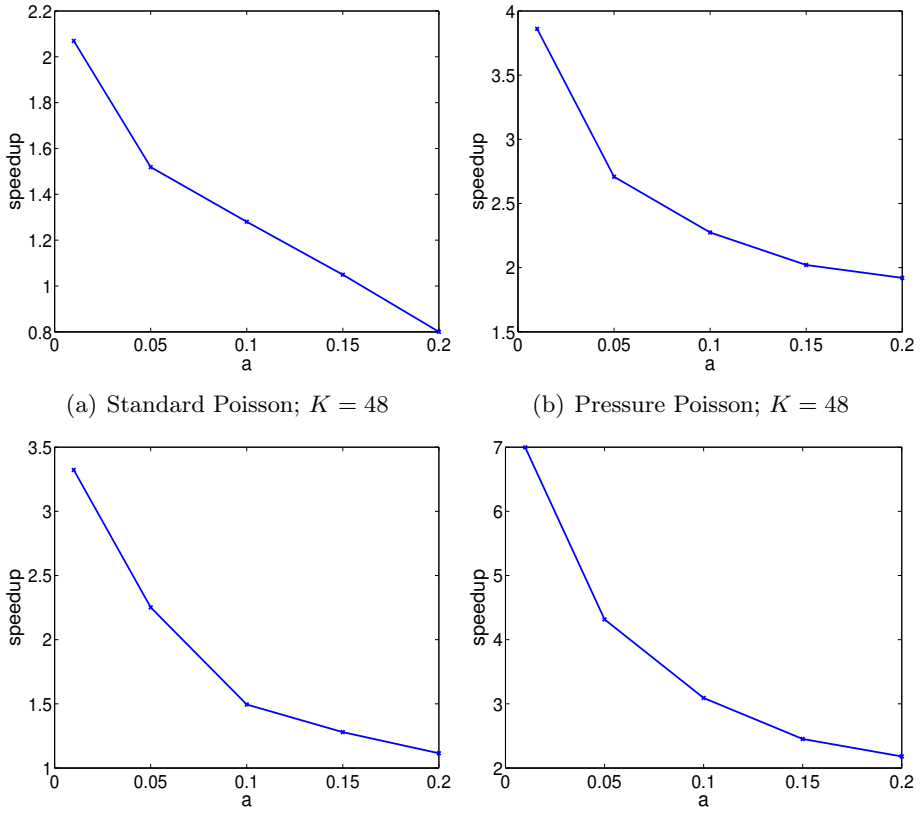
(c) Standard Poisson; $K = 480$

(d) Pressure Poisson; $K = 480$

Figure 15: Speedup as a function of the taper ratio γ when using the tensor-product solver as a preconditioner in a conjugate gradient iteration. In (a) and (b) we use $K = 48$ spectral elements, each of order $N = 16$. In (c) and (d) we use $K = 480$ spectral elements, each of order $N = 4$. We compare the elapsed cpu time relative to the 3D reference solver [7].

The second geometry we consider is a standard (straight) cylinder with a bump at the bottom. This geometry is of interest in studying how such a deformation might affect the pattern formation in Bénard-Marangoni convection [13]. Our bump is a Gaussian with a standard deviation set to span approximately 4 elements of our grid; see Figure 14. We consider the speedup as a function of the amplitude a of this Gaussian, where a is measured in percentage of the container height. The results for the two grid configurations are given in Figure 16.

We observe a rather rapid decline in the speedup for the standard Laplace operator. While the consistent pressure Poisson operator also experiences decline, it is still quite efficient, even with an amplitude being 20% of the container height. For the application we have in mind the geometry deformations are expected to be quite small. Hence, the results are quite encouraging for our intended application.



(a) Standard Poisson; $K = 48$

(b) Pressure Poisson; $K = 48$

(c) Standard Poisson; $K = 480$

(d) Pressure Poisson; $K = 480$

Figure 16: Speedup as a function of the amplitude a of the deformation when using the tensor-product solver as a preconditioner in a conjugate gradient iteration. In (a) and (b) we use $K = 48$ spectral elements, each of order $N = 16$. In (c) and (d) we use $K = 480$ spectral elements, each of order $N = 4$. We compare the elapsed cpu time relative to the 3D reference solver [7].

9 Summary and conclusions

We have presented a new tensor-product algorithm for solving linear systems of equations in partially deformed three-dimensional geometries. In particular, we have extended the method presented in [1] to solve incompressible fluid flow problems. The focus has been on solving the consistent pressure Poisson problem arising in the context of solving the unsteady Stokes or Navier-Stokes equations using mixed finite/spectral elements in combination with a pressure-velocity splitting approach.

Certain conditions need to be fulfilled in order to use the new solution algorithm: (i) the computational domain must have a two-dimensional cross-section which is invariant in the third direction; and (ii) the boundary conditions for the first two velocity components (i.e., the components associated with the cross-sectional plane) must be of the same *type* (either Dirichlet or Neumann).

The new method requires the solution of a series of independent two-dimensional systems. The discrete operator associated with these systems is generally not the same as the two-dimensional version of the consistent pressure Poisson operator, but is rather an operator which resembles a "Helmholtz-like" pressure operator. In the context of iterative solution

of the independent two-dimensional systems, we have proposed extensions of currently available preconditioners for the consistent pressure Poisson operator to this new operator. An advantage of the current method is that it is generally more efficient to solve several two-dimensional systems instead of a single three-dimensional system. The new method is also relatively easy to implement and offers alternative ways to exploit parallel architectures; this will be explored in a separate paper.

In the single-processor context, we have achieved speedup between 4 and 20 compared to an alternative three-dimensional reference solver. The preconditioner used to solve the independent two-dimensional systems can be regarded as an extention and adaptation of the preconditioner used in the reference solver. Hence, if other alternative reference solvers are used, extensions and adaptations of these can also be used to solve the independent two-dimensional systems. In this respect, we believe the speedup observed in this work should also extend to alternative realizations of the new tensor-product solver and to alternative reference solvers.

If the assumption of an "extruded" geometry is only approximately satisfied, the proposed solver may also be used as a global preconditioner in a conjugate gradient iteration. We have tested this idea for two different domains: a tapered cylinder and a cylindrical container with a deformation along the bottom. The new solver offers speedup as long as the geometry is not too far from an ideal situation.

10 Acknowledgment

The work has been supported by the Norwegian University of Science and Technology and the Research Council of Norway under contract 159553/I30 and contract 10323100. The support is gratefully acknowledged.

References

[1] T. Bjøntegaard, Y. Maday, and E. M. Rønquist. Fast tensor-product solvers: Partially deformed three-dimensional domains. *J. Sci. Comput.*, 39:28–48, 2009.

[2] F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *Rev. Francaise Automat. Informat. Recherche Operationelle Ser Rouge*, 8, 1974.

[3] M.S. Carvalho and L.E. Scriven. Three-dimensional stability analysis of free surface flows: application to forward deformable roll coating. *J. Comput. Phys.*, 151:534–562, 1999.

[4] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comput*, 22, 1968.

[5] D. Chu, R. Henderson, and G. E. Karniadakis. Parallel spectral-element-Fourier simulation of turbulent flow over riblet-mounted surfaces. *Theoretical and Computational Fluid Dynamics*, 3(4):219–229, 1992.

[6] M. Dryja and O. B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In Tony Chan, Roland Glowinski, Jacques Périaux, and Olof Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 3–21. SIAM, Philadelphia, PA, 1990.

[7] P.F. Fischer. An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 133:84–101, 1997.

[8] K. Goda. A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *J. Comput. Phys.*, 30, 1979.

[9] W. J. Gordon and C. A. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *Intl. J. Numer. Meth. Eng*, 7:461–477, 1973.

[10] J. L. Guermond and J. Shen. On the error estimates for the rotational pressure-correction projection methods. *Math. Comp*, 73:1719–1737, 2003.

[11] J. L. Guermond, J. Shen, and P. Minev. An overview of projection methods for incompressible flows. *Comp. Met. Mech. Eng.*, 2005.

[12] R. Henderson. Nonlinear dynamics and pattern formation in turbulent wake transition. *J. Fluid Mech.*, 353:65–112, 1997.

[13] E. L. Koschmieder. *Bénard Cells and Taylor Vortices*. Cambridge University Press, 1993.

[14] A. M. Kvaring, T. Bjøntegaard, and E. M. Rønquist. On pattern selection in three-dimensional Bénard-Marangoni flows. *Submitted to Communications in Computational Physics*, 2010.

[15] A. M. Kvarving. A fast tensor-product solver for incompressible fluid flow in partially deformed three-dimensional domains: Parallel implementation. *In preparation*, 2010.

[16] J. W. Lottes and P. F. Fischer. Hybrid Multigrid/Schwarz Algorithms for the Spectral Element Method. *J. Sci. Comput.*, 24, 2005.

[17] R. E. Lynch, J. R. Rice, and D. H. Thomas. Direct solution of partial difference equations by tensor product methods. *Numer. Math.*, 6:185–199, 1964.

[18] Y. Maday, D. Meiron, A. T. Patera, and E. M. Rønquist. Analysis of iterative methods for the steady and unsteady Stokes problem: Application to spectral element discretizations. *J. Sci. Comput.*, 14:310–337, 1993.

[19] Y. Maday and A. T. Patera. Spectral element methods for the incompressible Navier-Stokes equations. In *State-of-the-art surveys on computational mechanics (A90-47176 21-64). New York, American Society of Mechanical Engineers*, pages 71–143, 1989.

[20] Y. Maday, A. T. Patera, and E. M. Rønquist. The $P_N \times P_{N-2}$ method for the approximation of the Stokes problem. Technical Report 92009, Department of Mechanical Engineering, Massachusetts Institute of Technology, 1992.

[21] M. Medale and P. Cerisier. Numerical simulation of Bénard-Marangoni convection in small aspect ratio containers. *Numerical Heat Transfer, Part A*, 42:55–72, 2002.

[22] V. D. Narasimhamurthy, H. I. Andersson, and B. Pettersen. Cellular vortex shedding behind a tapered circular cylinder. *Physics of Fluids*, 21(4):044106, 2009.

[23] E. M. Rønquist. A Domain Decomposition Method for Elliptic Boundary Value Problems: Application to Unsteady Incompressible Fluid Flow. *Fifth conference on Domain Decompositions Methods for Partial Differential Equations (symposium held in Norfolk, Virginia, May 1991)*, pages 545–557, 1992.

[24] R. Temam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des par fractionnaires ii. *Arch. Ration. Metch. Anal*, 33:377–385, 1969.

[25] L. J. P. Timmermans, P. D. Minev, and F. N. Van De Vosse. An approximate projection scheme for incompressible flow using spectral elements. *Intl. J. Numer. Met. Fluids*, 22, 1996.

[26] A. Toselli and O. B. Widlund. *Domain Decomposition Methods - Algorithms and Theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer, 2004.

[27] H.M. Tufo and P.F. Fischer. Terascale spectral element algorithms and implementations. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, page 68, New York, NY, USA, 1999. ACM.

[28] J. van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *J. Sci. Stat. Comput.*, 3, 1986.