Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 300 (2016) 129–145
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Parameter estimates for the Relaxed Dimensional Factorization preconditioner and application to hemodynamics

Michele Benzi[a], Simone Deparis[b], Gwenol Grandperrin[b,*], Alfio Quarteroni[b]

[a] *Department of Mathematics and Computer Science, Emory University, Atlanta, GA 30322, USA*
[b] *MATHICSE - Chair of Modelling and Scientific Computing (CMCS), EPFL, CH - 1015 Lausanne, Switzerland*

Received 30 March 2015; received in revised form 4 November 2015; accepted 11 November 2015
Available online 2 December 2015

Highlights

- Relaxed Dimensional Factorization preconditioner (RDF).
- Algebraic estimation technique for finding a suitable value of the RDF parameter in both the 2D and the 3D case with arbitrary geometries.
- Application to approximation of the Navier–Stokes equations using a Marker-and-Cell scheme and a finite element one.
- Testing on large-scale problems relevant for hemodynamics simulation using up to 8196 cores.

Abstract

We present new results on the Relaxed Dimensional Factorization (RDF) preconditioner for solving saddle point problems from incompressible flow simulations, first introduced in Benzi et al. (2011). This method contains a parameter $\alpha > 0$, to be chosen by the user. Previous works provided an estimate of α in the 2D case using Local Fourier Analysis. A novel algebraic estimation technique for finding a suitable value of the RDF parameter in both the 2D and the 3D case with arbitrary geometries is proposed. This technique is tested on a variety of discrete saddle point problems arising from the approximation of the Navier–Stokes equations using a Marker-and-Cell scheme and a finite element one. We also show results for a large-scale problem relevant for hemodynamics simulation that we solve in parallel using up to 8196 cores.

Keywords: Scalable parallel preconditioners; Finite element method; High performance computing; Navier–Stokes equations; Hemodynamics applications; Dimensional splitting preconditioner

1. Introduction

In the last decade, many techniques have been proposed for preconditioning linear systems of equations in saddle point form, like those arising from the discretization of the Navier–Stokes equations. Among the most successful,

* Corresponding author.
E-mail addresses: benzi@mathcs.emory.edu (M. Benzi), simone.deparis@epfl.ch (S. Deparis), gwenol.grandperrin@gmail.com (G. Grandperrin), alfio.quarteroni@epfl.ch (A. Quarteroni).

http://dx.doi.org/10.1016/j.cma.2015.11.016
0045-7825/

we mention the Pressure Convection–Diffusion (PCD) preconditioner [1–3], which makes the GMRES iterations converge with a rate independent of the mesh, at least when the viscosity is sufficiently large.

An alternative to PCD is represented by the so-called Least-Squares Commutator (LSC) preconditioner [4,5,3], which can be built automatically, albeit with higher computational cost. The convergence rate of LSC is independent of the mesh size and mildly dependent on the viscosity. A version for stabilized finite element discretizations has been introduced in [5].

In [6], an Augmented Lagrangian (AL) preconditioner is introduced starting from the augmented Lagrangian formulation of the underlying saddle point problem. The corresponding convergence rate is independent of the mesh size [7], mildly dependent on the viscosity, and robust when anisotropic meshes are considered.

In [8], the Modified Augmented Lagrangian (MAL) preconditioner is introduced to make the action of the AL method cheaper and easier to implement, particularly on unstructured grids. When using the MAL preconditioner, the rate of convergence shows a mild dependence on the viscosity and is independent of the mesh size [7]. Like the AL preconditioner, MAL is robust when anisotropic meshes are used [9].

More recently, the so-called Relaxed Dimensional Factorization (RDF) preconditioner has been introduced in [10] as an improvement to the Dimensional Splitting (DS) preconditioner [11]. Although this method was mainly intended to solve steady Stokes and Oseen problems, it can also handle the unsteady case. Experimental results indicate independence of its convergence rate of the mesh size and a mild dependence on the viscosity. This preconditioner relies on a parameter that, in simple 2D geometries, can be estimated using Local Fourier Analysis (LFA). A comparison of the performance of the RDF preconditioner and other preconditioners such as PCD or LSC can be found in [10]. Those results showed that RDF can be an attractive alternative to PCD and LSC, especially for low values of the viscosity and anisotropic meshes. In this paper, we develop a new approach to estimating the RDF parameter for both 2D and 3D problems in arbitrary geometries. We test our technique on a few numerical benchmarks, as well as on a large 3D problem originating from the simulation of blood flow in large arteries, see e.g. [12]. In particular, we investigate the performance of the preconditioner in terms of strong scalability using up to 8192 cores.

The remainder of this paper is organized as follows. The mathematical model and the strategy to solve saddle point problems using the RDF preconditioner are presented in Sections 2 and 3, respectively. In Section 4, we propose a new technique to estimate the parameter of the RDF preconditioner. Then in Section 5, we test the RDF preconditioner on simple 2D cases, on a 3D driven cavity problem, and on a benchmark relevant to hemodynamic simulations using up to 8192 cores. Finally, some conclusions are drawn in Section 6.

2. Mathematical model

We consider an incompressible Newtonian fluid with constant density ρ and viscosity μ in a bounded domain Ω of $\mathbb{R}^d$ ($d = 2, 3$). The Navier–Stokes equations read

$$\frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = \mathbf{f} \qquad \text{in } \Omega,\ t > t_0, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega,\ t > t_0, \tag{2}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ is the velocity vector field, $p = (\mathbf{x}, t)$ the pressure scalar field, $\mathbf{f}$ the external force per mass unit, and $\nu = \frac{\mu}{\rho}$ the kinematic viscosity. These partial differential equations are complemented with an initial solution and boundary conditions:

$$\mathbf{u}(\mathbf{x}, t_0) = \mathbf{u}_0(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega,$$

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{g}_D(\mathbf{x}, t) \qquad \forall \mathbf{x} \in \Gamma_D,\ t > t_0, \tag{3}$$

$$\left(\nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - p\mathbf{n}\right)(\mathbf{x}, t) = \mathbf{g}_N(\mathbf{x}, t) \qquad \forall \mathbf{x} \in \Gamma_N,\ t > t_0, \tag{4}$$

where Γ_D and Γ_N refer to the Dirichlet and Neumann part of the boundary, respectively, $\Gamma_D \cup \Gamma_N = \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$, and $\mathbf{u}_0$, $\mathbf{g}_D$, and $\mathbf{g}_N$ are assigned functions.

We consider a fully implicit scheme, see, e.g. [13,14], to discretize in time the equations:

$$\frac{1}{\Delta t}\mathbf{u}^{n+1} - \nu \Delta \mathbf{u}^{n+1} + \mathbf{u}^{n+1} \cdot \nabla \mathbf{u}^{n+1} + \nabla p^{n+1} = \mathbf{f}^{n+1} + \frac{1}{\Delta t}\mathbf{u}^{n},$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0.$$

Picard iterations are used to solve the nonlinearity; see, e.g., [13]. The weak formulation of the resulting equations is discretized in space using the Finite Element Method (FEM) yielding at each time step a linear system of the form $\mathcal{A}_{NS}\mathbf{x} = \mathbf{b}$ with

$$\mathcal{A}_{NS} = \begin{pmatrix} F_1 & 0 & 0 & B_1^T \\ 0 & F_2 & 0 & B_2^T \\ 0 & 0 & F_3 & B_3^T \\ -B_1 & -B_2 & -B_3 & 0 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} U_1^{n+1} \\ U_2^{n+1} \\ U_3^{n+1} \\ P^{n+1} \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} G_1 \\ G_2 \\ G_3 \\ 0 \end{pmatrix}. \tag{5}$$

Here F_i $(i = 1, 2, 3)$ are discrete operators corresponding to the three components of the vector convection–diffusion–reaction operator

$$\frac{1}{\Delta t} - \nu\Delta + \mathbf{v} \cdot \nabla$$

acting on $\mathbf{u}^{n+1}$. Here $\mathbf{v}$ is the current Picard approximation to the velocity field $\mathbf{u}$, and B_i is a discrete approximation of the partial derivative $\frac{\partial}{\partial x_i}$, $(i = 1, 2, 3)$, so that $[B_1\ B_2\ B_3]^T$ is the discrete gradient and $-[B_1\ B_2\ B_3]$ is its adjoint, namely, the discrete divergence. Note that we are using $-B_i$ instead of B_i, $i = 1, 2, 3$; this choice guarantees that all the eigenvalues of $\mathcal{A}_{NS}$ lie in the right half-plane [15,16]. Finally, the G_i $(i = 1, 2, 3)$ contain the discretized source forces and the second part of the time derivatives which depend on $\mathbf{u}^n$. Both the matrix and the right hand side of the system are modified to take the boundary conditions into account.

3. Relaxed Dimensional Factorization (RDF) preconditioner

The Relaxed Dimensional Factorization (RDF) preconditioner was introduced in [10] as an improved version of the Dimensional Splitting (DS) preconditioner [11]. It was originally designed for steady Oseen problems with small viscosity ν, possibly using anisotropic meshes. On such problems, most of the preconditioners fail to converge at all or converge very slowly as showed in [10]. The RDF preconditioner exploits the structure of the matrix of the linear system (5) and reads:

$$\mathcal{P}_{RDF} = \frac{1}{\alpha^2} \begin{pmatrix} F_1 & 0 & 0 & B_1^T \\ 0 & \alpha I & 0 & 0 \\ 0 & 0 & \alpha I & 0 \\ -B_1 & 0 & 0 & \alpha I \end{pmatrix} \begin{pmatrix} \alpha I & 0 & 0 & 0 \\ 0 & F_2 & 0 & B_2^T \\ 0 & 0 & \alpha I & 0 \\ 0 & -B_2 & 0 & \alpha I \end{pmatrix} \begin{pmatrix} \alpha I & 0 & 0 & 0 \\ 0 & \alpha I & 0 & 0 \\ 0 & 0 & F_3 & B_3^T \\ 0 & 0 & -B_3 & \alpha I \end{pmatrix},$$

where $\alpha > 0$ is a parameter to be chosen. By expanding the product, we can observe that $\mathcal{P}_{RDF}$ coincides with $\mathcal{A}_{NS}$ up to additional terms (which depend on α) showing up on the upper triangular part:

$$\mathcal{P}_{RDF} = \begin{pmatrix} F_1 & -\frac{1}{\alpha}B_1^T B_2 & -\frac{1}{\alpha}B_1^T B_3 & B_1^T \\ 0 & F_2 & -\frac{1}{\alpha}B_2^T B_3 & B_2^T \\ 0 & 0 & F_3 & B_3^T \\ -B_1 & -B_2 & -B_3 & \alpha I \end{pmatrix}. \tag{6}$$

Numerical experiments conducted in [10] on 2D problems using $\mathbb{Q}_2-\mathbb{Q}_1$ and $\mathbb{Q}_2-\mathbb{P}_1$ FE on structured grids show that the RDF preconditioner leads to better results than the DS preconditioner for Stokes problems, Oseen and generalized Oseen problems, when the dynamic viscosity ν is relatively small. For unsteady problems, the convergence rate of RDF preconditioned iterations is independent of h and ν. Experiments with inexact variants of the RDF preconditioner also indicate h-independent convergence rates. However, a moderate dependency on ν is noticed. In [10], it is shown that RDF is generally more robust and effective than PCD, in particular for small ν. It should be mentioned, however, that RDF is more expensive than PCD to build.

We also point out that the RDF preconditioner bears some resemblance with a class of preconditioners known as constraint preconditioners [17]. Indeed, from (6) we see that the preconditioner is of the form

$$\mathcal{P}_{RDF} = \begin{pmatrix} G & B^T \\ -B & \alpha I \end{pmatrix},$$

where $B = [B_1\ B_2\ B_3]$ and G corresponds to the leading block 3×3 submatrix in (6). For small α this matrix becomes close to

$$\begin{pmatrix} G & B^T \\ -B & 0 \end{pmatrix},$$

which is a constraint preconditioner (meaning that the preconditioner contains the same constraint as the original coefficient matrix, in this case an incompressibility constraint).

4. Estimates for the parameter of the RDF preconditioner

When the RDF preconditioner is used in combination with GMRES, the rate of convergence is not overly sensitive with respect to the parameter α, as pointed out in [10]; for a value of α close enough to the optimal value, the convergence rate remains acceptable. The optimal value α_{opt} of α is the one that minimizes the number of GMRES iterations and was determined in [10] by LFA for 2D problems; this technique is recalled in Section 4.1 of this paper. For 3D problems, empirical search was used to find a good estimate for α_{opt}. In Section 4.2 we introduce a new parameter selection strategy for both 2D and 3D problems based on an analysis of the trace of the preconditioned matrix.

4.1. Local Fourier analysis for the determination of α_{opt}

LFA is a classical tool for parameter estimation in iterative methods, see, e.g., [5,18,19]. In particular, estimates for the α parameter of the RDF preconditioner in the context of a fluid cavity problem have been found using this technique in [10] in the 2D case. This type of analysis is based on the following assumptions:

1. the viscosity ν and $\boldsymbol{\beta}$ in the convective term $\boldsymbol{\beta} \cdot \nabla$ are constant;
2. periodic boundary conditions are imposed;
3. centered finite differences are used to discretize the problem;
4. the discrete problem is extended to an infinite uniform structured grid;
5. F_1, F_2, B_1, B_2, are all square of the same order and commute.

The first assumption may seem quite restrictive. However, we note that when ν is large the diffusion term $-\nu\Delta$ dominates the convection term $\mathbf{v} \cdot \nabla$. The Laplacian term is also dominating the convection term for h small enough. Indeed, for finite difference discretization the entries of the stiffness matrix associated to the Laplacian scale as $\mathcal{O}(h^{-2})$ and the entries of the matrix associated to the convection as $\mathcal{O}(h^{-1})$. Under these assumptions, F_1, F_2, are replaced by discrete unsteady convection–diffusion operators of the form

$$\frac{1}{\Delta t} I + \nu L_x + N_x, \qquad \frac{1}{\Delta t} I + \nu L_y + N_y,$$

where I is the identity operator, L_x, and L_y are discrete one-dimensional Laplacians obtained by centered differences and N_x, and N_y are the one-dimensional convection operators obtained by centered differences in the x, y, and z directions respectively. B_1, B_2 are replaced with the one-dimensional differentiation operators S_x and S_y obtained by one-sided differences in x, and y directions, respectively. Finally, to mimic the scaling of the entries of the finite element mass matrix, we multiply these operators by h^2 in 2D. For the sake of comparison, we assume, as in [10], that $\Delta t \approx h$. The symbols corresponding to the operators are the following

$$\begin{aligned}
\frac{1}{\Delta t} I + \nu L_x + N_x &: \nu(1 - e^{i2\pi h\theta_x} - e^{-i2\pi h\theta_x}) + h(1 + \beta_{\max,x}(e^{i2\pi h\theta_x} - e^{-2p\pi h\theta_x})) \\
\frac{1}{\Delta t} I + \nu L_y + N_y &: \nu(1 - e^{i2\pi h\theta_y} - e^{-i2\pi h\theta_y}) + h(1 + \beta_{\max,y}(e^{i2\pi h\theta_y} - e^{-2p\pi h\theta_y})) \\
S_x &: h(1 - e^{-i2\pi h\theta_x}) \\
S_y &: h(1 - e^{-i2\pi h\theta_y}),
\end{aligned}$$

where $\beta_{\max,x}$ and $\beta_{\max,y}$ denote the maximum of the x and y components of the convective field $\boldsymbol{\beta}$ (i.e., to compute an approximation to α, we perform a Fourier analysis using a constant field $\boldsymbol{\beta}$ taken as the maximum velocity observed in the considered problem). For arbitrary meshes, we choose h to be the average of the diameter of the tetrahedra.

In this case, it has been shown in [10] that a good estimate for α_{opt} is given by

$$\alpha_* = \arg\min_{\alpha} |\mu(\alpha) - 1|, \tag{7}$$

where $\mu(\alpha)$ is defined as follows:

$$\mu(\alpha) = \frac{1}{\alpha}(s_1 + s_2) - \frac{2}{\alpha^2} s_1 s_2. \tag{8}$$

Here s_1 and s_2 are the eigenvalues of $S_1 = B_1(F_1 + \frac{1}{\alpha} B_1^T B_1)^{-1} B_1^T$ and $S_2 = B_2(F_2 + \frac{1}{\alpha} B_2^T B_2)^{-1} B_2^T$, respectively. We observe that the quantity $\mu(\alpha)$ is to be minimized not only with respect to α but also with respect to all the frequencies θ_x, θ_y that appear in the expression for the eigenvalues of S_1 and S_2; we refer to [10] for details. In practice, the computational cost for computing α_* for a given value of h and ν is small compared to the solution of the saddle point problem, and the computation can be performed off-line (i.e., one can precompute the value of α a priori and then use this value in further computations) for a broad range of values of h and ν.

When a FE method is used to discretize the equations, the entries of the mass matrix M_u scale as $\mathcal{O}(h^{-2})$. Therefore, we expect small values for the parameter α.

A major limitation of LFA is that it is not clear how it can be used in the case of complex, irregular geometries and unstructured meshes. In the following we introduce a purely algebraic technique that can be used to estimate the optimal α regardless of the underlying geometry or the discretization scheme used.

4.2. Estimation using matrix traces

Experience shows that the number of preconditioned GMRES iterations is usually lower when the eigenvalues of the right preconditioned matrix $\mathcal{T}_\alpha = \mathcal{A}_{NS}\mathcal{P}_{RDF}^{-1}$ (or, equivalently, of the left preconditioned matrix $\mathcal{T}'_\alpha = \mathcal{P}_{RDF}^{-1}\mathcal{A}_{NS}$) are clustered around one. For this reason, we propose to use the following value for α:

$$\alpha_{trace} = \arg\min_{\alpha} \left| \sum_i (\lambda_i - 1) \right| = \arg\min_{\alpha} |\operatorname{Tr}(\mathcal{T}_\alpha) - N|,$$

where the λ_i's denote the eigenvalues of $\mathcal{T}_\alpha$ (and $\mathcal{T}'_\alpha$), and N is the rank of $\mathcal{A}_{NS}$. As proved in [10] all these eigenvalues have positive real part. Computing the trace of $\mathcal{T}_\alpha$ (or of $\mathcal{T}'_\alpha$, which is the same) is an expensive task. To reduce its cost, we use an explicit expression of $\mathcal{T}'_\alpha$ that is provided by Lemma 1.

Lemma 1. *Let* $\hat{F}_i = F_i + \frac{1}{\alpha} B_i^T B_i$ *and* $S_i = B_i \hat{F}_i^{-1} B_i^T$, $i = 1, 2, 3$. *Then*

$$\mathcal{T}'_\alpha = I - \mathcal{P}_{RDF}^{-1}\mathcal{R}_\alpha = I - \begin{pmatrix} 0 & -\frac{1}{\alpha}\hat{F}_1^{-1}B_1^T B_2 & -\frac{1}{\alpha}\hat{F}_1^{-1}B_1^T B_3 & -\hat{F}_1^{-1}B_1^T \\ 0 & \frac{1}{\alpha^2}\hat{F}_2^{-1}B_2^T S_1 B_2 & -\frac{1}{\alpha^2}\hat{F}_2^{-1}B_2^T(\alpha I - S_1)B_3 & -\frac{1}{\alpha}\hat{F}_2^{-1}B_2^T(\alpha I - S_1) \\ 0 & \frac{1}{\alpha^3}\hat{F}_3^{-1}B_3^T(\alpha I - S_2)S_1 B_2 & \frac{1}{\alpha^3}\hat{F}_3^{-1}B_3^T[(\alpha I - S_2)S_1 + \alpha S_2]B_3 & -\frac{1}{\alpha^2}\hat{F}_3^{-1}B_3^T(\alpha I - S_2)(\alpha I - S_1) \\ 0 & -\frac{1}{\alpha^4}(\alpha I - S_3)(\alpha I - S_2)S_1 B_2 & -\frac{1}{\alpha^4}(\alpha I - S_3)[(\alpha I - S_2)S_1 + \alpha S_2]B_3 & \frac{1}{\alpha^3}(\alpha I - S_3)(\alpha I - S_2)(\alpha I - S_1) \end{pmatrix}.$$

Proof. Let $\mathcal{P}_{RDF} = \frac{1}{\alpha^2}\mathcal{M}_1\mathcal{M}_2\mathcal{M}_3$ with

$$\mathcal{M}_1 = \begin{pmatrix} F_1 & 0 & 0 & B_1^T \\ 0 & \alpha I & 0 & 0 \\ 0 & 0 & \alpha I & 0 \\ -B_1 & 0 & 0 & \alpha I \end{pmatrix}, \qquad \mathcal{M}_2 = \begin{pmatrix} \alpha I & 0 & 0 & 0 \\ 0 & F_2 & 0 & B_2^T \\ 0 & 0 & \alpha I & 0 \\ 0 & -B_2 & 0 & \alpha I \end{pmatrix},$$

$$\mathcal{M}_3 = \begin{pmatrix} \alpha I & 0 & 0 & 0 \\ 0 & \alpha I & 0 & 0 \\ 0 & 0 & F_3 & B_3^T \\ 0 & 0 & -B_3 & \alpha I \end{pmatrix}, \quad \text{then we have}$$

$$\mathcal{M}_1^{-1} = \begin{pmatrix} \hat{F}_1^{-1} & 0 & 0 & -\frac{1}{\alpha}\hat{F}_1^{-1}B_1^T \\ 0 & \frac{1}{\alpha}I & 0 & 0 \\ 0 & 0 & \frac{1}{\alpha}I & 0 \\ \frac{1}{\alpha}B_1\hat{F}_1^{-1} & 0 & 0 & \frac{1}{\alpha}I - \frac{1}{\alpha^2}S_1 \end{pmatrix}, \qquad \mathcal{M}_2^{-1} = \begin{pmatrix} \frac{1}{\alpha}I & 0 & 0 & 0 \\ 0 & \hat{F}_2^{-1} & 0 & -\frac{1}{\alpha}\hat{F}_2^{-1}B_2^T \\ 0 & 0 & \frac{1}{\alpha}I & 0 \\ 0 & \frac{1}{\alpha}B_2\hat{F}_2^{-1} & 0 & \frac{1}{\alpha}I - \frac{1}{\alpha^2}S_2 \end{pmatrix},$$

$$\mathcal{M}_3^{-1} = \begin{pmatrix} \frac{1}{\alpha}I & 0 & 0 & 0 \\ 0 & \frac{1}{\alpha}I & 0 & 0 \\ 0 & 0 & \hat{F}_3^{-1} & -\frac{1}{\alpha}\hat{F}_3^{-1}B_3^T \\ 0 & 0 & \frac{1}{\alpha}B_3\hat{F}_3^{-1} & \frac{1}{\alpha}I - \frac{1}{\alpha^2}S_3 \end{pmatrix}.$$

Therefore,

$$\mathcal{P}_{RDF}^{-1} = \begin{pmatrix} \hat{F}_1^{-1} & 0 & 0 & -\frac{1}{\alpha}\hat{F}_1^{-1}B_1^T \\ -\frac{1}{\alpha}\hat{F}_2^{-1}B_2^T B_1\hat{F}_1^{-1} & \hat{F}_2^{-1} & 0 & -\frac{1}{\alpha^2}\hat{F}_2^{-1}B_2^T(\alpha I - S_1) \\ -\frac{1}{\alpha^2}\hat{F}_3^{-1}B_3^T(\alpha I - S_2)B_1\hat{F}_1^{-1} & -\frac{1}{\alpha}\hat{F}_3^{-1}B_3^T B_2\hat{F}_2^{-1} & \hat{F}_3^{-1} & -\frac{1}{\alpha^3}\hat{F}_3^{-1}B_3^T(\alpha I - S_2)(\alpha I - S_1) \\ \frac{1}{\alpha^3}(\alpha I - S_3)(\alpha I - S_2)B_1\hat{F}_1^{-1} & \frac{1}{\alpha^2}(\alpha I - S_3)B_2\hat{F}_2^{-1} & \frac{1}{\alpha}B_3\hat{F}_3^{-1} & \frac{1}{\alpha^4}(\alpha I - S_3)(\alpha I - S_2)(\alpha I - S_1) \end{pmatrix}.$$

Finally, direct computations give the result.

We now state the following result:

Lemma 2. *The trace of $\mathcal{T}_\alpha$ and $\mathcal{T}'_\alpha$ is given by*

$$\mathrm{Tr}(\mathcal{T}_\alpha) = \mathrm{Tr}(\mathcal{T}'_\alpha) = N_{\mathbf{u}} + \frac{1}{\alpha}\mathrm{Tr}(S_1 + S_2 + S_3) - \frac{2}{\alpha^2}\mathrm{Tr}(S_1S_2 + S_1S_3 + S_2S_3) + \frac{2}{\alpha^3}\mathrm{Tr}(S_1S_2S_3),$$

where $S_i = B_i\hat{F}_i^{-1}B_i^T$, $i = 1, 2, 3$ and $N_{\mathbf{u}}$ is the number of degrees of freedom for the discrete velocity.

Proof. It is obvious that $\mathcal{T}_\alpha$ and $\mathcal{T}'_\alpha$ have the same trace. From Lemma 1, we find

$$\begin{aligned}\mathrm{Tr}(\mathcal{T}'_\alpha) = \mathrm{Tr}(I) &- \frac{1}{\alpha^2}\mathrm{Tr}(\hat{F}_2^{-1}B_2^T S_1 B_2) - \frac{1}{\alpha^3}\mathrm{Tr}\left(\hat{F}_3^{-1}B_3^T\left((\alpha I_p - S_2)S_1 + \alpha S_2\right)B_3\right) \\ &- \frac{1}{\alpha^3}\mathrm{Tr}\left((\alpha I_p - S_3)(\alpha I_p - S_2)(\alpha I_p - S_1)\right).\end{aligned}$$

Thanks to the property $\mathrm{Tr}(XY) = \mathrm{Tr}(YX)$ for $X \in \mathbb{R}^{N\times M}$ and $Y \in \mathbb{R}^{M\times N}$, simple computations lead to

$$\mathrm{Tr}(\hat{F}_2^{-1}B_2^T S_1 B_2) = \mathrm{Tr}(B_2\hat{F}_2^{-1}B_2^T S_1), \quad \text{and}$$
$$\mathrm{Tr}\left(\hat{F}_3^{-1}B_3^T\left((\alpha I_p - S_2)S_1 + \alpha S_2\right)B_3\right) = \mathrm{Tr}\left(B_3\hat{F}_3^{-1}B_3^T\left((\alpha I_p - S_2)S_1 + \alpha S_2\right)\right),$$

and this concludes the proof.

The computation of $\mathrm{Tr}(S_i)$, $\mathrm{Tr}(S_iS_j)$ or $\mathrm{Tr}(S_1S_2S_3)$ for $i, j = 1, 2, 3$ in Lemma 2 is the most expensive part of this approach and we would like to avoid it. More generally, computing or estimating the trace of a product of matrices or the trace of the inverse of a matrix are challenging problems with no easy solution. To simplify the computation, we approximate $\hat{F}_i = F_i + \frac{1}{\alpha}B_i^T B_i$ by F_i and S_i by $\tilde{S}_i := B_i\mathrm{diag}(F_i)^{-1}B_i^T$. Then, we can explicitly form $\tilde{S}_i$ for $i = 1, 2, 3$ and find α_{trace} by minimizing the function

$$\phi(\alpha) := N_{\mathbf{u}} + \frac{1}{\alpha}a - \frac{2}{\alpha^2}b + \frac{2}{\alpha^3}c,$$

where

$$a := \mathrm{Tr}(\tilde{S}_1 + \tilde{S}_2 + \tilde{S}_3), \qquad b := \mathrm{Tr}(\tilde{S}_1\tilde{S}_2 + \tilde{S}_1\tilde{S}_3 + \tilde{S}_2\tilde{S}_3), \qquad c := \mathrm{Tr}(\tilde{S}_1\tilde{S}_2\tilde{S}_3)$$

are independent of α. To further reduce the cost of this computation, we make use of the two following formulas valid for $A, B \in \mathbb{R}^{N\times N}$:

$$\begin{aligned} \mathrm{Tr}(A+B) &= \mathrm{Tr}(A) + \mathrm{Tr}(B), \\ \mathrm{Tr}(AB) &= \sum_{i,j=1}^{N} (A \circ B^T)_{ij}, \end{aligned} \tag{9}$$

where $\circ$ denotes the Hadamard product.

Finally, we observe that it is inexpensive to evaluate $\phi(\alpha)$ once a, b, and c have been computed. Indeed, one can simply evaluate the function for α in a given range to find an approximation of the minimum value, i.e., α_{trace}.

5. Numerical results

We now focus on testing the efficiency of the RDF preconditioner. In particular, we test the techniques that were introduced in Section 4 to estimate α_{opt}. The numerical experiments are split into two parts. The first one shows some Matlab experiments that allow us to compare how the new estimates for α compete against the Fourier analysis estimate introduced in [10]. The second one shows some numerical experiments using up to 8192 cores on the aneurysm benchmark problem introduced in [20].

5.1. Three simple benchmark tests

In this section, we want to compare the iterations count using the new estimates with those obtained using the parameter (7) yielded by Fourier Analysis (FA). We consider the same settings as in [10]: we solve the Oseen equations using two test problems. First, we consider the steady 2D lid-driven cavity problem discretized by $\mathbb{Q}_2 - \mathbb{P}_1$ finite elements, which satisfy the Babuška–Brezzi (inf–sup) condition [21], on uniform grids. In particular, we consider three values for the viscosity $\nu = 0.1$, $\nu = 0.01$, and $\nu = 0.005$, and four different space discretizations corresponding to 16×16, 32×32, 64×64, and 128×128 structured meshes. Next, we consider a 3D Marker-and-Cell (MAC) discretization of the Oseen problem [22]. All the simulations are carried out with Matlab [23]. The FE discretizations are generated using the IFISS 3.0 software package [24].

Unless otherwise specified, the linear system arising from the Oseen problem is solved using a right preconditioned restarted GMRES; the maximum subspace dimensions are set to 20. We use a zero initial guess and the stopping criterion is based on the norm of the residual scaled by the right hand side, i.e.,

$$\frac{\|\mathbf{r}\|_2}{\|\mathbf{b}\|_2} \leq 10^{-6}.$$

For each problem, we report the number of iterations for different choices of α stemming from the strategies presented in Section 4, as well as with the optimal value obtained experimentally by an expensive "brute force search". In all the examples, the subproblems involved in the application of the RDF preconditioner are solved by direct methods, since our first goal is to determine the effectiveness of the new parameter estimation techniques. Note that we are only interested in a good choice for α to lower the number of iterations. Hence, we only report the iterations count for GMRES. We refer to [10] for numerical experiments involving the use of subiterations to apply $\hat{F}_i^{-1}$, $i = 1, 2, 3$, instead of direct methods.

5.1.1. The 2D leaky lid driven cavity problem discretized by $\mathbb{Q}_2 - \mathbb{P}_1$ finite elements

This test problem describes the behavior of a fluid in a cavity and driven by a current at the top (the lid) of the domain. For example, one can think of a cavity in the bed of a river where the fluid starts moving due to the river flowing at the top. The computational domain Ω is the square $[-1, 1]^2$. The x component of the velocity u_x is imposed on the upper side of the square such that $u_x = 1$. No-slip boundary conditions are imposed on the lower, left and right sides.

Table 1
Preconditioned GMRES on steady Oseen problems with different grid sizes ($\mathbb{Q}_2 - \mathbb{P}_1$ FEM, uniform grids), viscosity $\nu = 0.1$.

Grid	16×16		32×32		64×64		128×128	
	α	Iter	α	Iter	α	Iter	α	Iter
RDF optimal	0.13	8	0.04	8	0.011	8	0.005	7
RDF LFA estimate	0.073	9	0.025	8	0.008	8	0.002	7
RDF trace estimate	0.119	8	0.032	8	0.008	8	0.002	7

Table 2
Preconditioned GMRES on steady Oseen problems with different grid sizes ($\mathbb{Q}_2 - \mathbb{P}_1$ FEM, uniform grids), viscosity $\nu = 0.01$.

Grid	16×16		32×32		64×64		128×128	
	α	Iter	α	Iter	α	Iter	α	Iter
RDF optimal	0.55	12	0.17	11	0.041	10	0.011	10
RDF LFA estimate	0.126	21	0.057	17	0.024	13	0.010	10
RDF trace estimate	1.188	15	0.321	14	0.083	12	0.021	11

Table 3
Preconditioned GMRES on steady Oseen problems with different grid sizes ($\mathbb{Q}_2 - \mathbb{P}_1$ FEM, uniform grids), viscosity $\nu = 0.005$.

Grid	16×16		32×32		64×64		128×128	
	α	Iter	α	Iter	α	Iter	α	Iter
RDF optimal	0.70	15	0.22	14	0.06	13	0.017	13
RDF LFA estimate	0.127	32	0.062	25	0.028	17	0.012	13
RDF trace estimate	2.377	24	0.642	23	0.167	22	0.042	19

Table 4
Preconditioned GMRES on 3D steady Oseen problems with different grid sizes (MAC discretization, uniform grids), viscosity $\nu = 0.1$.

Grid	$16 \times 16 \times 16$		$32 \times 32 \times 32$	
	α	Iter	α	Iter
RDF optimal	2.2	11	2.3	12
RDF trace estimate	1.265	13	1.306	13

Tables 1–3 show the number of GMRES iterations to converge. The reported numbers are the maximum number of GMRES iterations required in any of the Picard steps. When $\nu = 0.1$ (see Table 1) we observe that the LFA and the strategy based on trace estimates provide essentially optimal results, and that the number of iterations remains independent of h. The results for $\nu = 0.01$ are reported in Table 2. As the mesh is refined, the quality of the LFA estimate improves, and indeed the LFA estimate performs quite well when sufficiently fine meshes are used (see also [10] for similar observations). Overall, the new trace-based method introduced in this work performs better than the LFA estimates. Finally, the results for $\nu = 0.005$ are reported in Table 3. The number of iterations obtained using the trace estimate is higher than for the larger values of ν but remains moderate and decreases as the mesh is refined. For this value of ν, the LFA estimate results in the lowest iteration count. All in all, the strategy based on the trace estimate works about as well as the one based on LFA estimate.

5.1.2. The 3D leaky lid driven cavity problem discretized by the Marker-and-Cell method

We now consider again a leaky lid driven cavity problem but in 3D and using Marker-and-Cell (MAC) discretization [22]. Tables 4–6 show the number of GMRES iterations to converge. For $\nu = 0.1$, $\nu = 0.01$, and $\nu = 0.005$, the trace estimate leads to essentially optimal results. Note that the optimal α is much larger for the MAC discretization than for the FE one, reflecting the different scaling of the discretized equations resulting from the two schemes.

Table 5
Preconditioned GMRES on 3D steady Oseen problems with different grid sizes (MAC discretization, uniform grids), viscosity $\nu = 0.01$.

Grid	$16 \times 16 \times 16$		$32 \times 32 \times 32$	
	α	Iter	α	Iter
RDF optimal	16.9	17	13.7	18
RDF trace estimate	12.651	18	13.064	18

Table 6
Preconditioned GMRES on 3D steady Oseen problems with different grid sizes (MAC discretization, uniform grids), viscosity $\nu = 0.005$.

Grid	$16 \times 16 \times 16$		$32 \times 32 \times 32$	
	α	Iter	α	Iter
RDF optimal	26.7	24	26.5	24
RDF trace estimate	25.982	24	26.133	24

5.2. Large-scale experiments

For large-scale 3D problems, exact application of the RDF preconditioner is too expensive and inexact variants must be used [10]. The approximation of the RDF preconditioner uses the factorization of the RDF preconditioner from Section 3 as a starting point. Here, inverses of the algebraic operators are replaced by some approximations denoted by a "tilde" over the corresponding operators, which we describe below. The approximate RDF preconditioner can then be defined in factored form, as follows:

$$
\begin{aligned}
\mathcal{P}_{aRDF}^{-1} = &
\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & \frac{1}{\alpha}B_3 & I \end{pmatrix}
\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & \tilde{\hat{F}}_3^{-1} & 0 \\ 0 & 0 & 0 & I \end{pmatrix}
\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & -B_3^T \\ 0 & 0 & 0 & I \end{pmatrix} \\
&
\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & \frac{1}{\alpha}B_2 & 0 & I \end{pmatrix}
\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & \tilde{\hat{F}}_2^{-1} & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}
\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & -\frac{1}{\alpha}B_2^T \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & \frac{1}{\alpha}I \end{pmatrix} \\
&
\begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ B_1 & 0 & 0 & I \end{pmatrix}
\begin{pmatrix} \tilde{\hat{F}}_1^{-1} & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}
\begin{pmatrix} I & 0 & 0 & -\frac{1}{\alpha}B_1^T \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix},
\end{aligned}
$$

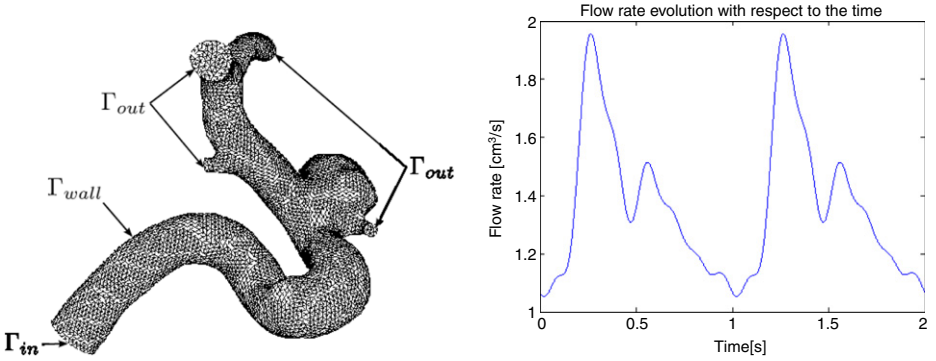
where $\tilde{\hat{F}}_i$ is an approximation of $\hat{F}_i = F_i + \frac{1}{\alpha}B_i^T B_i$ for $i = 1, 2, 3$.

In our experiments we use GMRES with a 2-level additive Schwarz preconditioner to approximate the action of $\hat{F}_i^{-1}$ for $i = 1, 2, 3$ (i.e., we iterate until the inverse is applied accurately enough). More in particular, we create a partition from the full domain. Each parallel task performs computation on one of the subdomains. Additive-Schwarz (1 level) with minimal overlap (one layer of elements) is used as a pre-smoother. An LU factorization is used for the (exact) solve on each subdomain. No post-smoothing is applied. The coarse grid is obtained algebraically and the solve at the coarse level is performed using one sweep of the Gauss–Seidel method. Note that applying Additive Schwarz with minimal overlap (equivalent to a block Jacobi preconditioner) avoids communications. This choice has a positive impact on the cost to compute the preconditioner since as the number of parallel tasks is increasing, the size N of the matrix of the local problem is decreasing, and hence the LU factorization computational cost is reduced as well.

The three-dimensional simulations have been coded in `LifeV` [25], a C++ finite element library under the LGPL license. This library makes intensive use of `Trilinos` [26], which contains many distributed packages. In particular, the multigrid algorithm relies on the `ML` package, and GMRES is based on the `Belos` package. Whenever mesh

Table 7
Monte Rosa Cray XE6 technical data.

Number of service nodes	40
Number of compute nodes	1,496
Number of processors per node	2x 16-core AMD Opteron Interlagos
Processors frequency	2.1 GHz
Processors shared memory per node	32 GB DDR2
Peak performance	402 Teraflop/s.
Network	Gemini 3D torus



(a) Mesh of the cerebral aneurysm.

(b) Flux evolution with respect to the time.

Fig. 1. Mesh and inflow for the cerebral aneurysm.

coarsening is required, aggregates are computed using `METIS/ParMETIS` [27,28]. To perform the LU factorization required by the coarse solve of multigrid, we chose to use KLU [29] rather than UMFPACK [30]. Although KLU has been developed primarily for solving sparse linear systems from circuit simulations and not the ones arising in finite element modeling, we found that the time to compute the factorization is smaller than that of UMFPACK and this makes the total costs (time to build the preconditioners and time for the preconditioned iterations) smaller. All the computations are carried out using Monte Rosa, a Cray XE6 supercomputer at the Swiss National Supercomputing Centre (CSCS), cf. Table 7. The reported results are obtained using as many cores as possible on each node, without multithreading enabled.

5.2.1. Flow in a cerebral aneurysm

We consider the simulation of blood flow in a cerebral aneurysm, which is a localized blood-filled deformation in a blood vessel wall. The computational domain Ω represents an artery where an aneurysm has developed (see Fig. 1(a)). The geometry of the cerebral aneurysm was first used in the research project Aneurisk [31]. Detailed information on the meshes is provided in Table 9. The diameter of the inlet Γ_{in} measures 0.35 cm and is chosen as characteristic length. Fig. 1(a) shows the mesh used to perform the simulations, for a time step $\Delta t = 10^{-3}$. Blood flow is modeled by the Navier–Stokes equations with density $\rho = 1$ g/cm^3 and viscosity $\mu = 0.035$ g/(cm s), i.e., $\nu = 0.035$ cm^2/s. An approximation of the flow rate at the inlet Γ_{in} has been provided in [32] as

$$\varphi(t) = a_0 + \sum_{k=1}^{7} a_k \cos\left(\frac{2\pi k t}{T}\right) + b_k \sin\left(\frac{2\pi k t}{T}\right), \tag{10}$$

where T denotes the period of the cardiac cycle, and a_k and b_k are given in Table 8 (the coefficients have been scaled to correspond to our geometry). The function $\varphi(t)$ is presented in Fig. 1(b). For our computations, we take $T = 1$ and the flow rate is imposed by a flat inlet profile on Γ_{in}. This choice is made for simplicity only; using more "physiological" boundary conditions (see [12] for a discussion) would not impact the performance of the preconditioner. Homogeneous Neumann boundary conditions are imposed on Γ_{out}, and no-slip boundary conditions are imposed on the vessel wall Γ_{wall}.

Table 8
Coefficients for the flow rate function, [32].

	0	1	2	3	4	5	6	7
a_k	1.36	−0.207	−0.152	0.059	0.039	0.00286	−0.0371	−0.000759
b_k		0.176	−0.0429	−0.117	0.0385	0.0139	0.0166	−0.0359

Table 9
Aneurysm test case: Number of Degrees of Freedom (DoF) and mesh size.

Mesh	Velocity DoF	Pressure DoF	h_{min}	$h_{average}$	h_{max}
Coarse	597,093	27,242	0.015	0.035	0.059
Medium	4,557,963	199,031	0.005	0.018	0.051
Fine	35,604,675	1,519,321	0.0026	0.0097	0.0277

The problem is discretized in time using a fully implicit scheme linearized with Picard–Oseen iterations with $\Delta t = 10^{-3}$ and in space using the $\mathbb{P}_2 - \mathbb{P}_1$ finite elements on a tetrahedral mesh. Using larger time steps was found to result in the Picard iteration failing to converge in some cases. The tolerance ϵ_{NL} for the nonlinear iterations of the implicit scheme is set to 10^{-6}. Flexible GMRES (FGMRES) without restart is used to solve the linear problem at each time step [33]. The preconditioner is recomputed at each time iteration. The stopping criterion is based on the residual scaled by the right hand side,

$$\|\mathbf{r}_k\|_2 \leq 10^{-6} \|\mathbf{b}\|_2 .$$

We evaluate four versions of the approximate RDF preconditioner: each of them corresponds to the tolerance that is used to apply $\hat{F}_i^{-1}$, $i = 1, 2, 3$ using the inner AMG-preconditioned GMRES method, namely 10^{-1}, 10^{-2}, 10^{-4}, and 10^{-10}. For this reason, at each Picard iteration, the system is solved using FGMRES. Using high accuracy in the inner solves is of course very expensive and can be avoided, as the results will confirm in what follows. We include results for the 10^{-10} tolerance to show that less accuracy in the inner solves does not cause the convergence of the outer FGMRES iteration to deteriorate. In practice, using a large tolerance of 10^{-1} results in the fastest solution time, about 10 times faster than using 10^{-10}. We first discuss the efficacy of the strategy based on the trace estimation technique. Table 10 shows the approximate value for α_{opt} computed by successive trials, and the value of α_{trace} obtained using the trace estimation technique. The results are obtained using 1024 processes and where $\hat{F}_i^{-1}$, $i = 1, 2, 3$ have been applied with a tolerance of 10^{-10}. Using the medium mesh, we see that we are close to α_{opt}. We have not computed an estimation for α_{opt} using the fine mesh to save resources and since the iteration count is already even better than for the medium mesh. We observe that the values of α_{opt} are now much smaller than those reported for the simple benchmark problems. While we do not have a complete explanation for this fact, we note that already in [10] it was observed that the optimal α is smaller for unsteady problems than for steady ones, and that lower values of ν and h lead to smaller values of α_{opt}.

Fig. 2 gives more details about how the iterations count behaves with respect to α using the coarse and the medium meshes. The average number of Picard iterations to solve one time step depends on the choice of the mesh; using the coarse, the medium, and the fine mesh, we observed 6, 4, and 4 Picard iterations to converge to the solution, respectively. In our experiment, the inner tolerance did not affect the number of Picard iterations.

We first report the time to build the RDF preconditioner in Fig. 3. With the coarse mesh, strong scalability is observed up to 512 cores. With the medium and fine meshes, the time is superlinear, thanks to the LU factorization computed on each subdomain as explained at the beginning of Section 5.2.

We report in Table 11 the time needed to compute α_{trace} (in parenthesis, we report the fraction w.r.t. the total time), the time to compute $\hat{F}_1$ and the total time to build the RDF preconditioner. With all the meshes, we note that computing α_{trace} does not scale at all with an increasing number of processors. In fact, the time is even increasing. However, the value of α_{trace} does not vary much during the simulation. Therefore, in practice, one can compute it once and for all using a lower number of cores than the number used to do the simulations, at a negligible cost. The computation does not scale because the entries of the matrices B and B^T are stored by rows, which are owned by different parallel tasks. Therefore, when we compute the product as in Eq. (9), a lot of communication is required to transpose the matrix, i.e., the rows of the matrix become the columns and the row entries have to be sent to the appropriate parallel task.

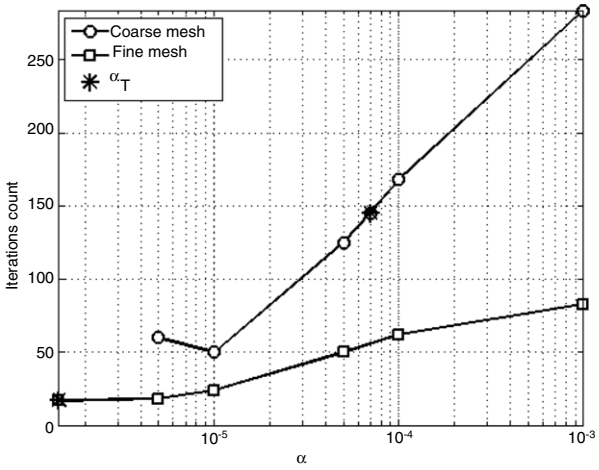
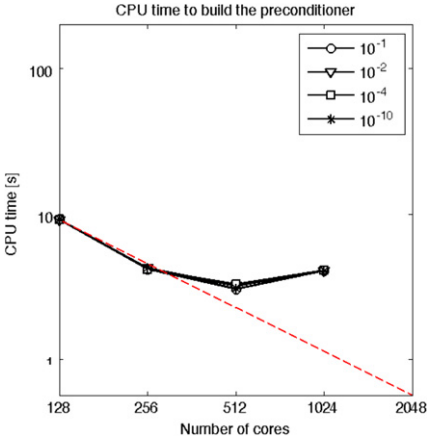
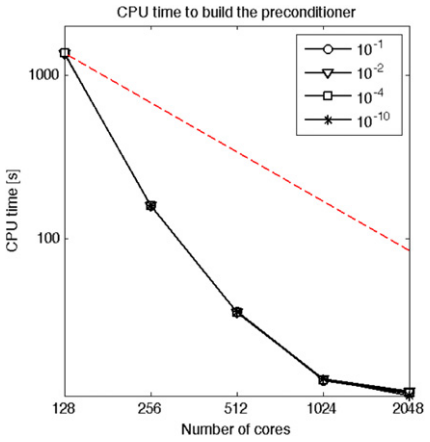


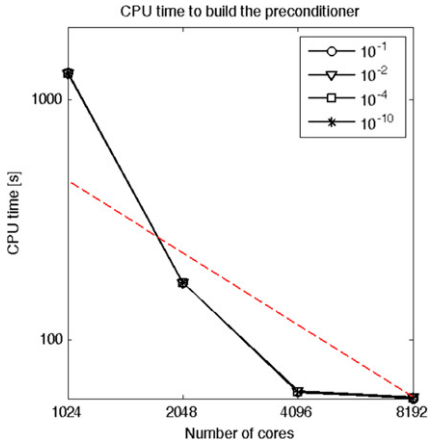
Fig. 2. Aneurysm test case: number of GMRES iterations with respect to the choice of α.



(a) Coarse mesh.



(b) Medium mesh.



(c) Fine mesh.

Fig. 3. Aneurysm test case: CPU time needed to build the preconditioners.

Table 10
Aneurysm test case: comparison between α and α_{opt}. Inner tolerance $= 10^{-10}$.

Mesh	α_{opt}	Iteration count for α_{opt}	α_{trace}	Iteration count for α_{trace}
Coarse	$\sim 10^{-5}$	50	$7 \cdot 10^{-5}$	137.33 (6)
Medium	$\sim 5 \cdot 10^{-6}$	16.4	$1.5 \cdot 10^{-6}$	17.25 (4)
Fine	–	–	$1.76 \cdot 10^{-6}$	9.75 (4)

Table 11
Aneurysm test case: time to build the RDF preconditioner.

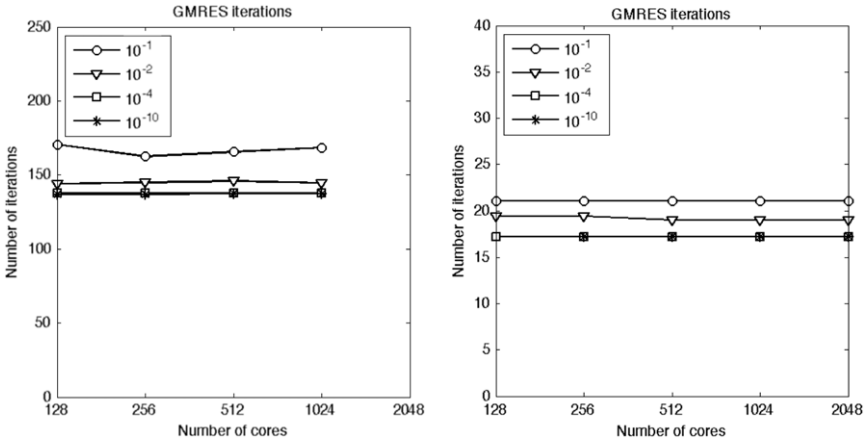
Mesh	Cores	Computation of α_{trace}		Preconditioner for $\hat{F}_1$	Total time
Coarse	128	1.09	(0.12)	2.61	9.13
	256	0.9	(0.22)	1.0	4.1
	512	1.0	(0.34)	0.59	2.95
	1024	1.72	(0.43)	0.64	3.97
Medium	128	4.77	$(3.57 \cdot 10^{-3})$	268.3	1336.85
	256	3.16	(0.02)	52.0	158.73
	512	2.48	(0.07)	10.74	35.33
	1024	2.57	(0.19)	3.42	13.49
	2048	4.29	(0.36)	1.91	11.89
Fine	1024	6.97	$(5.38 \cdot 10^{-3})$	240.12	1296.21
	2048	7.18	(0.04)	54.48	172.55
	4096	13.0	(0.22)	14.47	60.34
	8192	23.22	(0.41)	9.41	56.25

Building the RDF preconditioner is fairly expensive, because the number of entries contained in the matrices $\hat{F}_i$, $i = 1, 2, 3$ compared to those of the matrix F_i, is high. We examined a sparse version of $\hat{F}_i$ where only the entries that are already in the pattern of F are kept. Therefore, the number of nonzero entries of the sparse $\hat{F}_i$ is the same as that of F. Unfortunately, the resulting preconditioned FGMRES method exhibits poor convergence, although this version of the RDF preconditioner can be built very quickly. Future research should be aimed at finding a version that leads to good convergence rates of FGMRES using sparse approximations of $\hat{F}_i$, $i = 1, 2, 3$. In this direction, we mention a promising variant of the augmented Lagrangian approach where a Grad–Div stabilization technique is used to build the augmented matrix; see [34,35]. In this approach the augmentation is performed at the continuous level, before discretization. Experiments show that this reduces the number of entries while preserving the convergence rates. A similar approach may work well in the context of RDF preconditioning.

Fig. 4 reports the number of iterations of FGMRES. With the coarse mesh the number of iterations remains constant for the range of number of cores that we considered. Nevertheless, the iterations count is quite large for all the precisions with which we apply the RDF. It is possible that this is due to the mesh being too coarse to yield physically meaningful solutions, and to the onset of instabilities ("wiggles") in the corresponding discrete solution. Using the medium or the fine mesh, however, the number of iterations is much lower than with the coarse mesh and remains constant. We note that applying less accurately the RDF preconditioner slightly increases the number of iterations, but in all the considered cases, the number of iterations remains almost constant.

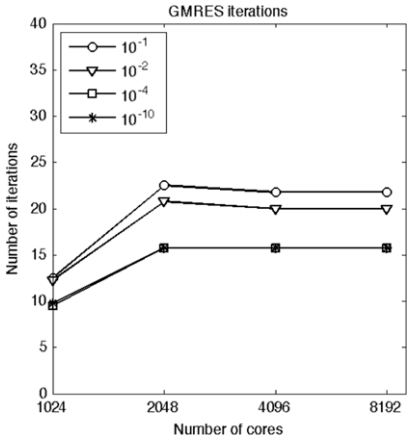
Finally, we show the scalability curves of the time needed to solve the linear system (FGMRES iterations) in Fig. 5. With the coarse mesh the strong scalability is observed up to 512 cores, while the scalability is almost perfect with the medium and fine mesh. The accuracy used to apply the RDF preconditioner is here again not affecting the slope of the scalability curves. Therefore, it is suitable to apply the inexact version using the inner tolerance 0.1, which as already mentioned is about 10 times faster than when using the tolerance 10^{-10}. We report the average number of iterations and the time to solve the system at each Picard iteration in Table 12. Overall, the RDF preconditioner, while expensive to build, is very robust and is able to make the iterative method converge to the solutions even for very difficult cases.

To summarize, the RDF preconditioner built using α computed with the trace estimate leads to a robust and scalable preconditioning technique. The main issue remains the relatively high computation time to build it due to the number of nonzero entries in $\hat{F}_i$, $i = 1, 2, 3$. Therefore, finding a sparse approximation to $\hat{F}_i$, $i = 1, 2, 3$,



(a) Coarse mesh.

(b) Medium mesh.



(c) Fine mesh.

Fig. 4. Aneurysm test case: number of FGMRES iterations.

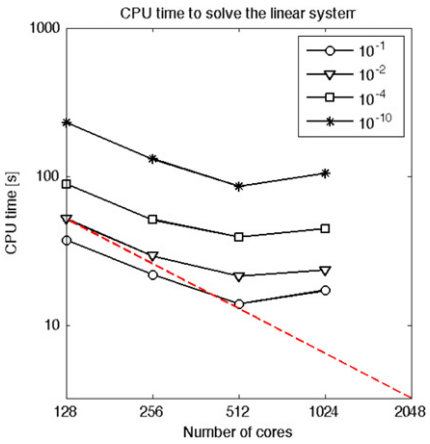
will be the key to obtain a more efficient version of the RDF preconditioner. It should be mentioned that for this particular benchmark problem, numerical experiments described in [36] indicate that an approximate SIMPLE preconditioner (see, e.g., [15]) leads to faster overall solution times than RDF, mainly due to the lower cost of forming the preconditioner. Nevertheless, for large time steps or for steady problems the performance of SIMPLE is known to deteriorate. The purpose of these tests was mostly to assess the scalability of the method and the effectiveness of the trace-based parameter estimation technique on a realistic, large-scale test problem.

6. Conclusion

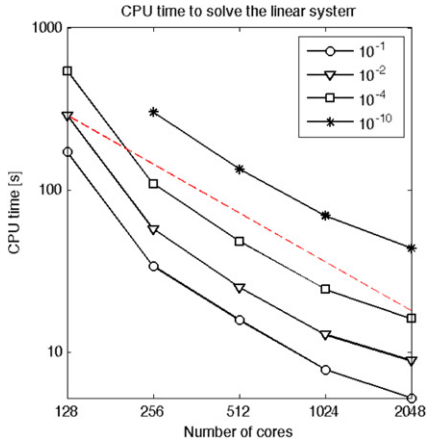
In this paper, we introduced a new technique to estimate the parameter α in the RDF preconditioner. Unlike LFA, the new approach (based on the trace of the preconditioned matrix) results in very good convergence rates for both 2D and 3D cases. Numerical experiments on a large 3D hemodynamics problem show that RDF scales very well with increasing core counts. Future work should focus on improving the efficiency of the preconditioner. In particular, techniques to reduce fill-in in the sub-matrices that comprise the RDF preconditioner should be investigated.

Acknowledgments

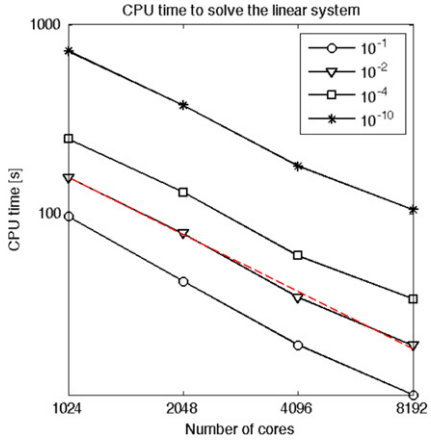
The work of M. Benzi was supported in part by Department of Energy (Office of Science) grant ERKJ247. G. Grandperrin was supported by the HP2C project lead by the Swiss National Supercomputer Centre (CSCS) in



(a) Coarse mesh.



(b) Medium mesh.



(c) Fine mesh.

Fig. 5. Aneurysm test case: CPU time for the preconditioned iterations.

Table 12
Aneurysm test case: time of the preconditioned iterations.

Mesh	Cores	Average iteration count	Time
Coarse	128	140.7	31.3
	256	135.2	18.3
	512	136.9	11.6
	1024	139.7	14.4
Medium	128	24.5	208.0
	256	23.3	37.8
	512	23.2	17.5
	1024	23.2	8.6
	2048	23.3	5.7
Fine	1024	13.2	106.4
	2048	22.3	44.2
	4096	21.6	20.2
	8192	21.7	11.1

Lugano, Switzerland. The support of the Swiss National Supercomputing Centre (CSCS) under project IDs h02 and s392 was allowed to carry on the simulations of this paper. We want to thank Prof. Vaidy S. Sunderam of the

Department of Mathematics and Computer Science at Emory University for facilitating this collaboration. Some preliminary parallel experiments were conducted using the Blue Gene/P system at EPFL. The financial support for CADMOS and the Blue Gene/P system is provided by the Canton of Geneva, Canton of Vaud, Hans Wilsdorf Foundation, Louis-Jeantet Foundation, University of Geneva, University of Lausanne and Ecole Polytechnique Fédérale de Lausanne. Finally, we are also thankful to Prof. Daniel Kressner for pointing us to some references about the approximations of traces of matrices.

References

[1] D. Silvester, H. Elman, D. Kay, A. Wathen, Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow, J. Comput. Appl. Math. 128 (1–2) (2001) 261–279. Numerical analysis 2000, Vol. VII, Partial differential equations. http://dx.doi.org/10.1016/S0377-0427(00)00515-X.

[2] D. Kay, D. Loghin, A. Wathen, A preconditioner for the steady-state Navier-Stokes equations, SIAM J. Sci. Comput. 24 (1) (2002) 237–256. http://dx.doi.org/10.1137/S106482759935808X.

[3] H.C. Elman, R.S. Tuminaro, Boundary conditions in approximate commutator preconditioners for the Navier-Stokes equations, Electron. Trans. Numer. Anal. 35 (2009) 257–280.

[4] H. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, Block preconditioners based on approximate commutators, SIAM J. Sci. Comput. 27 (5) (2006) 1651–1668. http://dx.doi.org/10.1137/040608817.

[5] H. Elman, V.E. Howle, J. Shadid, D. Silvester, R. Tuminaro, Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations, SIAM J. Sci. Comput. 30 (1) (2007) 290–311. http://dx.doi.org/10.1137/060655742.

[6] M. Benzi, M.A. Olshanskii, An augmented Lagrangian-based approach to the Oseen problem, SIAM J. Sci. Comput. 28 (6) (2006) 2095–2113. http://dx.doi.org/10.1137/050646421. (Electronic).

[7] M. Benzi, M.A. Olshanskii, Field-of-values convergence analysis of augmented Lagrangian preconditioners for the linearized Navier-Stokes problem, SIAM J. Numer. Anal. 49 (2) (2011) 770–788. http://dx.doi.org/10.1137/100806485.

[8] M. Benzi, M.A. Olshanskii, Z. Wang, Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations, Internat. J. Numer. Methods Fluids 66 (66) (2011) 486–508.

[9] M. Benzi, Z. Wang, A parallel implementation of the modified augmented Lagrangian preconditioner for the incompressible Navier–Stokes equations, Numer. Algorithms 64 (1) (2013) 73–84. http://dx.doi.org/10.1007/s11075-012-9655-x.

[10] M. Benzi, M. Ng, Q. Niu, Z. Wang, A relaxed dimensional factorization preconditioner for the incompressible Navier-Stokes equations, J. Comput. Phys. 230 (16) (2011) 6185–6202. http://dx.doi.org/10.1016/j.jcp.2011.04.001.

[11] M. Benzi, X.-P. Guo, A dimensional split preconditioner for Stokes and linearized Navier-Stokes equations, Appl. Numer. Math. 61 (1) (2011) 66–76. http://dx.doi.org/10.1016/j.apnum.2010.08.005.

[12] A. Quarteroni, M. Tuveri, A. Veneziani, Computational vascular fluid dynamics: problems, models and methods, Comput. Vis. Sci. 2 (4) (2000) 163–197. http://dx.doi.org/10.1007/s007910050039.

[13] H.C. Elman, D.J. Silvester, A.J. Wathen, Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics, in: Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2005.

[14] A. Quarteroni, Numerical models for differential problems, in: MS&A. Modeling, Simulation and Applications, vol. 8, Springer-Verlag, Italia, Milan, 2014.

[15] M. Benzi, G.H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numer. 14 (2005) 1–137. http://dx.doi.org/10.1017/S0962492904000212.

[16] A. Quarteroni, A. Valli, Domain decomposition methods for partial differential equations, in: Numerical Mathematics and Scientific Computation, The Clarendon Press - Oxford University Press, New York, 1999, Oxford Science Publications.

[17] C. Keller, N.I.M. Gould, A.J. Wathen, Constraint preconditioning for indefinite linear systems, SIAM J. Matrix Anal. Appl. 21 (4) (2000) 1300–1317. http://dx.doi.org/10.1137/S0895479899351805.

[18] M. Benzi, Z. Wang, Analysis of augmented Lagrangian-based preconditioners for the steady incompressible Navier-Stokes equations, SIAM J. Sci. Comput. 33 (2011) 2761–2784.

[19] R. Wienands, W. Joppich, Practical Fourier analysis for multigrid methods, in: Numerical Insights, vol. 4, Chapman & Hall/CRC, Boca Raton, FL, 2005, with 1 CD-ROM (Windows and UNIX).

[20] S. Deparis, G. Grandperrin, A. Quarteroni, Parallel preconditioners for the unsteady Navier-Stokes equations and applications to hemodynamics simulations, Comput. & Fluids 92 (0) (2014) 253–273. http://dx.doi.org/10.1016/j.compfluid.2013.10.034.

[21] F. Brezzi, On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers, Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge 8 (R-2) (1974) 129–151.

[22] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (12) (1965) 2182–2189. http://dx.doi.org/10.1063/1.1761178.

[23] MATLAB, Version 7.14.0.739 (R2012a), The MathWorks Inc., Natick, Massachusetts, 2012.

[24] H.C. Elman, A. Ramage, D.J. Silvester, IFISS, a Matlab toolbox for modelling incompressible flow, ACM Trans. Math. Software 33 (2007) http://dx.doi.org/10.1145/1236463.1236469.

[25] EPFL, Polimi, Emory, INRIA, LifeV User Manual, 2010. http://www.lifev.org.

[26] M.A. Heroux, R.A. Bartlett, V.E. Howle, R.J. Hoekstra, J.J. Hu, T.G. Kolda, R.B. Lehoucq, K.R. Long, R.P. Pawlowski, E.T. Phipps, A.G. Salinger, H.K. Thornquist, R.S. Tuminaro, J.M. Willenbring, A. Williams, K.S. Stanley, An overview of the trilinos project, ACM Trans. Math. Software 31 (3) (2005) 397–423. http://dx.doi.org/10.1145/1089014.1089021.

[27] G. Karypis, K. Schloegel, V. Kumar, METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Tech. Rep., Univ MN, 1998.
[28] G. Karypis, K. Schloegel, V. Kumar, ParMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering library, Tech. Rep, Univ MN, 2003.
[29] T.A. Davis, E. Palamadai Natarajan, Algorithms 907: KLU, a direct sparse solver for circuit simulation problems, ACM Trans. Math. Software 37 (3) (2010) 36.
[30] T.A. Davis, Algorithms 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method, ACM Trans. Math. Software 30 (2) (2004) 196–199. http://dx.doi.org/10.1145/992200.992206.
[31] L.M. Sangalli, P. Secchi, S. Vantini, A. Veneziani, A case study in exploratory functional data analysis: geometrical features of the internal carotid artery, J. Am. Stat. Assoc. 104 (485) (2009) 37–48. http://dx.doi.org/10.1198/jasa.2009.0002.
[32] H. Baek, M.V. Jayaraman, P.D. Richardson, G.E. Karniadakis, Flow instability and wall shear stress variation in intracranial aneurysms, J. R. Soc. Interface 7 (2010) 967–988.
[33] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (2) (1993) 461–469. http://dx.doi.org/10.1137/0914028.
[34] T. Heister, G. Lube, G. Rapin, On robust parallel preconditioning for incompressible flow problems, in: Numerical Mathematics and Advanced Applications 2009, Springer, 2010, pp. 443–450.
[35] T. Heister, A massively parallel finite element framework with application to incompressible flows (Ph.D. thesis), Georg-August-Universität Göttingen, 2011.
[36] G. Grandperrin, Parallel preconditioners for Navier-Stokes equations and fluid–structure interaction problems: application to hemodynamics (Ph.D. thesis), EPFL, 2013.