14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020
Virtual Congress: 11-15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

ON PRECONDITIONING VARIABLE POISSON EQUATION WITH EXTREME CONTRASTS IN THE COEFFICIENTS

Àdel Alsalti-Baldellou [1,2], F. Xavier Trias[1], Andrey Gorobets[3] and Assensi Oliva[1]

[1]Heat and Mass Transfer Technological Center, Technical University of Catalonia
C/ Colom 11, 08222 Terrassa (Barcelona), Spain; www.cttc.upc.edu
{adel.alsalti, francesc.xavier.trias, asensio.oliva}@upc.edu

[2]Termo Fluids SL
C/ Magí Colet 8, 08204 Sabadell (Barcelona), Spain; www.termofluids.com

[3] Keldysh Institute of Applied Mathematics
4A, Miusskaya Sq., Moscow 125047, Russia; www.keldysh.ru
andrey.gorobets@gmail.com

Key words: Poisson Equation, Variable Preconditioning, Multiphase Flow, Adaptive Mesh Refinement

Abstract. It is well known that the solution by means of iterative methods of very ill-conditioned systems leads to very poor convergence rates. In this context, preconditioning becomes crucial in order to modify the spectrum of the system being solved and improve the performance of the solvers. A proper balance between the reduction in the number of iterations and the overhead of the construction and application of the preconditioner needs to be sought to actually decrease the total execution time of the solvers. This is particularly important when considering variable coefficients matrices as, in general, its preconditioners will also be variable and need to be updated regularly at an affordable cost. In this work we present a family of variable preconditioners designed for the effective solution of variable Poisson equation with extreme contrasts in the coefficients, which represents a particularly challenging case as it translates into a variable and extremely ill-conditioned system arising in many situations such as with multiphase flows presenting high density ratios or in the presence of highly-stretched adaptive mesh refinements. Finally, the results of the numerical experiments performed are presented and discussed, confirming our preconditioners as extremely affordable, highly-parallelizable and easy-to-implement alternatives to the more standard (and usually unfeasible) preconditioners, still showing great improvements in the rate of convergence of the solvers without requiring the variable coefficients matrix to be explicitly rebuilt at each iteration.

1 INTRODUCTION

Under certain assumptions, divergence constraints follow from basic conservation principles, such as the conservation of mass or electrical charge. Such constraints lead to a Poisson equation for a sort of scalar potential whose solution is usually one of the most time-consuming and difficult to parallelize parts of scientific simulation codes. Consequently, the development and implementation of efficient and scalable Poisson solvers results a crucial task for simulations of physical phenomena belonging to a wide

variety of fields, as computational fluid dynamics (CFD), linear elasticity, electrostatics or oil reservoir modeling, among many others.

As should be expected, changes in the discretization or in the configuration of the physical system will result into different versions of the discrete Poisson equation, presenting (eventually very) different properties that will need to be carefully considered and profited by the solvers. In this sense, the aim of this paper is to propose a solver for the variable coefficients Poisson equation which, in its general form and defining the scalar fields $\alpha(\mathbf{r},t), \phi(\mathbf{r},t), \psi(\mathbf{r},t) \in \mathbb{R}$, reads:

$$\nabla \cdot (\alpha \nabla \phi) = \psi$$

and, after an appropriate discretization, results into:

$$\mathsf{MAG}\phi_h = \psi_h$$

where $\mathsf{A} = \mathrm{diag}(\alpha_h)$, M and G are the discrete divergence and gradient operators, and α_h, ϕ_h and ψ_h the discrete scalar fields α, ϕ and ψ.

More concretely, we will focus on the case where the *coefficients* field, α, presents very high contrasts along the physical domain. Two factors make this particularly challenging: firstly, as A changes in time, so does the matrix of the system, MAG, being then unaffordable the use of direct solvers due to their setup costs (apart from their excessive memory requirements). Secondly, as direct solvers are unusable, then the use of iterative methods becomes imperative. However, and as will be seen in the forthcoming sections, high contrasts in the coefficients immediately translates into very ill-conditioned systems, leading to very slow convergences.

The need for solvers adequately adapted to such versions of the Poisson equation is not new and, indeed, shared by many different fields such as multiphase flows, groundwater flows or oil reservoir simulations or semiconductor modeling (cf. [1], [2], [3], [4]). Although much previous work had already been done, the first two papers analyzing the impact of extreme contrasts in the coefficients appeared in 1999: in [3], Vuik et al. applied it to oil reservoir modeling, where extreme contrasts in the permeability of sand and shale layers are present (typically leading to permeability ratios of order 10^{-7}). On the other side, in [5], Graham and Hagger applied it to the development of additive Schwarz domain decomposition preconditioners on unstructured meshes for such cases. More recent contributions incorporate the use of deflation techniques [6, 7, 8], model order reduction [2, 8] and multigrid preconditioning [9].

Despite considering extreme contrasts, nearly all the proposals mentioned so far do not tackle (at least explicitly) the extra problem of having to work with a variable coefficients matrix, which poses a huge extra challenge as setup costs may easily make a solver prohibitively expensive in computational terms. In this work we present a set of preconditioners specially designed for this case and compatible with the conjugate gradient method (CG) and, consequently, with the rest of less restrictive Krylov subspace methods, such as GMRES, CGS, BICGSTAB... The main features of our proposal are:

- It does not require the full matrix MAG to be built explicitly and updated at each iteration.
- The update costs of our variable preconditioners are rather negligible.
- It gives rise to a family of new "adaptive" preconditioners based on already existing ones that generally result unaffordable in the variable case.

- The extra operations involved in our preconditioners (with respect to the existing preconditioners from which they arise) are highly parallelizable, scalable and easy to implement.

On the other side, given the never-ending growth of the computing capacity and the general incorporation of massively parallel accelerators by most modern supercomputers, it is worth to emphasize that the update of our preconditioners does not require any inter or intra-node communication, as the operation is completely local. This would allow the scientific codes to not be affected by slower latencies and to efficiently engage all the available devices, achieving better scalabilities and higher performances thanks to their increased memory bandwidths.

The rest of the paper is organized as follows: in Section 2 the framework from which our preconditioners are derived will be detailed, that is: firstly, Poisson equation and its discretization in the context of CFD will be given; secondly, some fundamental properties of the iterative methods of choice will be reviewed; thirdly, preconditioning techniques will be introduced. Afterwards, in Section 3, the preconditioners that we propose will be derived as suitable alternatives to other ones already existing and widely applied to the non-variable Poisson equation. Then, in Section 4 numerical results obtained from a realistic multi phase test case will be given and, finally, in Section 5 some concluding remarks will be made.

2 VARIABLE COEFFICIENTS POISSON EQUATION

As was already pointed, variable versions of the Poisson equation with large discontinuities on its coefficients may be found in many different areas. Thus, even though the development of our preconditioners will be based on incompressible multi phase flows, its extension to other fields will be almost direct.

2.1 Poisson equation in CFD

Let us consider an incompressible multi phase flow governed by:

$$\begin{aligned} \text{Navier-Stokes:} \quad & \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v} = \frac{1}{\rho}\left(\mu\Delta\mathbf{v} - \nabla p + \sigma\kappa\delta_\Gamma\hat{\mathbf{n}}\right) \\ \text{Incompressibility:} \quad & \nabla\cdot\mathbf{v} = 0 \end{aligned}$$

where the velocity-pressure coupling is solved by means of the well known fractional step method of Chorin [10] and Témam [11], whose application leads to the solution of a variable Poisson equation to obtain the pressure field, p, that will later be used to correct the predictor velocity, $\mathbf{v}^p$, and project it onto a divergence-free subspace:

$$\nabla\cdot\left(\frac{1}{\rho}\nabla p\right) = \frac{1}{\Delta t}\nabla\cdot\mathbf{v}^p$$

Applying a symmetry-preserving discretization of the system (cf. [12, 13, 14]) results into the following discrete version of the Navier-Stokes equations:

$$\Omega\frac{d\mathbf{v}_h}{dt} = -\mathsf{C}(\mathbf{v}_h)\mathbf{v}_h + \mathsf{N}\mathsf{D}\mathbf{v}_h - \mathsf{R}^{-1}\Omega\mathsf{G}p_h + \sigma\mathsf{K}\mathsf{G}\theta_h, \text{ with } \begin{cases} \text{Convective operator: } \mathsf{C}(\mathbf{v}_h) \\ \text{Diffusive operator: } \mathsf{D} \\ \text{Mesh volumes: } \Omega = \text{diag}(V_h) \\ \text{Level set marker: } \theta_h \\ \mathsf{R} = \text{diag}(\rho_h),\ \mathsf{N} = \text{diag}(\mu_h/\rho_h) \end{cases}$$

being K the discrete curvature (see [12]) and with the discrete gradient and divergence operators related by:

$$\mathsf{G} = -\Omega^{-1}\mathsf{M}^t \tag{1}$$

and, therefore, the discrete constant Laplace operator taking the form: $\mathsf{L} = -\mathsf{M}\Omega^{-1}\mathsf{M}^t$. Additionally, our discrete variable coefficients Poisson equation for pressure would then read:

$$\tilde{\mathsf{L}}p = \mathsf{M}v^p \tag{2}$$

where the discrete variable Laplace operator is defined as $\tilde{\mathsf{L}} \coloneqq \mathsf{M}\mathsf{R}^{-1}\mathsf{G}$ and, by virtue of Equation 1, can be expressed as $\tilde{\mathsf{L}} = -\mathsf{M}\mathsf{R}^{-1}\Omega^{-1}\mathsf{M}^t$. It would be then meaningful to consider a sort of new metric of the system, $\tilde{\Omega}$, accounting for the density and defined as $\tilde{\Omega} \coloneqq \Omega\mathsf{R}$, thus reaching:

$$\tilde{\mathsf{L}} = -\mathsf{M}\tilde{\Omega}^{-1}\mathsf{M}^t. \tag{3}$$

2.2 Poisson solvers

As was commented in the introduction, the use of iterative solvers results mandatory as direct solvers are not usable because of the characteristics of the system we aim to solve and the typical mesh sizes involved.

Among the iterative solvers, Krylov subspace methods play a fundamental role thanks to their good performance and to being highly parallelizable. For a detailed review of such methods the reader is referred to [15].

Bearing in mind that under most discretizations $\tilde{\mathsf{L}}$ is a symmetric negative-semidefinite matrix (being $\ker\tilde{\mathsf{L}} = \langle\mathbf{1}\rangle$), it becomes clear that Equation 2 can be easily changed of sign to make its coefficients matrix positive-semidefinite. Under such circumstances, CG should be the solver of choice given its robustness, computationally cheap iterations and small memory requirements. A suitable choice for non-symmetric discretizations of $\tilde{\mathsf{L}}$ could be, among many others, GMRES (see [15]).

A well known result to describe the rate of convergence of CG applied to a general linear system $Ax = b$ is given by the following bound:

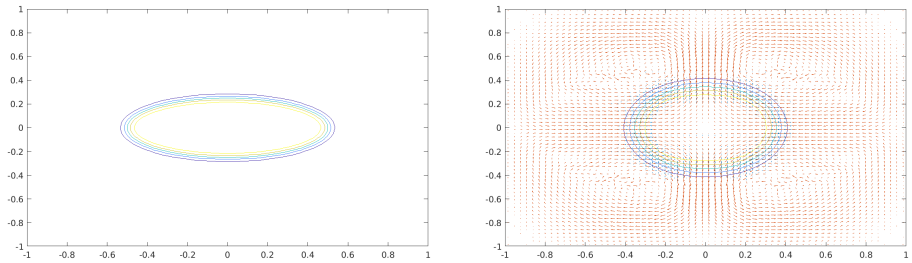
$$\|e_k\|_A \leq 2\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^k \|e_0\|_A\,, \text{ where } \kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)} \text{ and } \|e_k\|_A = \sqrt{(x_k - x_0)^t A\,(x_k - x_0)} \tag{4}$$

which sets a link between the rate of convergence of CG and the spectral properties of the coefficients matrix of the system, A, being λ_{max} and λ_{min} its largest and smallest (in magnitude) eigenvalues.

Roughly speaking we could say that: the higher the condition number of the coefficients matrix ($\kappa(A)$) is, the slower it will converge. This behavior is not exclusive with CG and similar bounds can be found for other iterative methods. Precisely for this, matrices having a large condition number are said to be ill-conditioned and expected to be harder to solve iteratively. Nevertheless, it is worth to mention that the convergence of CG (and the rest of iterative methods) can be a bit more complex, sometimes yielding to a superlinear convergence that can only be explained by means of the actual distribution of the eigenvalues. A deeper analysis of this topic focusing on the CG case can be found in [16].

Going back to the Poisson equation, it is interesting to illustrate the impact of extreme contrasts in the coefficients on the conditioning of the resulting system. With this aim, let us consider a two-fluid flow test case run on MATLAB that will also be used in future sections to confirm the expected behavior of our proposals. Our test case will be solved using a symmetry-preserving staggered discretization based on the results presented in [12].

More concretely, the initial static configuration shown in Figure 1a will be assumed, letting the surface tension make it evolve to the configuration shown in Figure 1b, to later perform all the numerical experiments on that final snapshot. Since this work strictly focuses on the solution of Equation 2, for the sake of simplicity the idealized parameters shown in Table 1 and a homogeneous spatial discretization have been considered, thus having: $\Omega = \Delta x \Delta y \mathbb{I}$ and $\tilde{\Omega} = \Delta x \Delta y \mathsf{R}$.



(a) Initial configuration (b) Evolved configuration

Figure 1: Two-fluid flow test case: surface tension acts on the initial static ellipse.

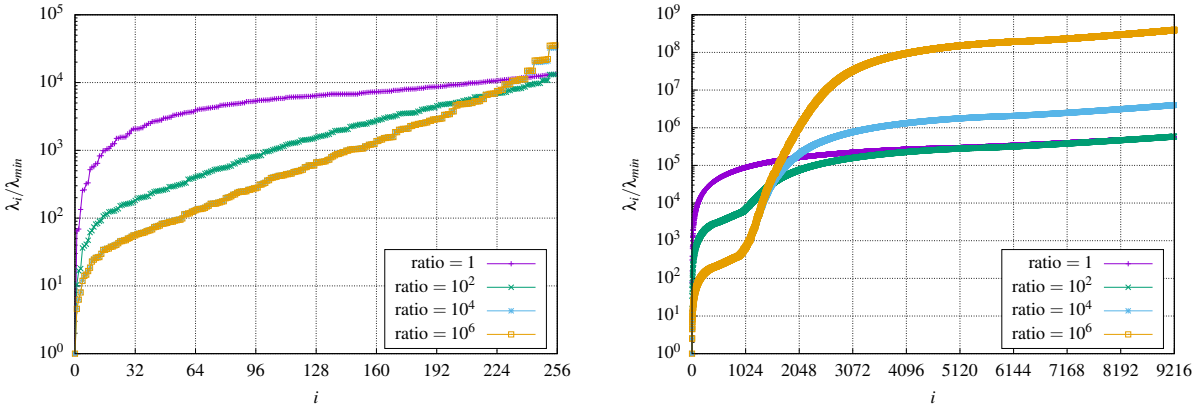
parameter		units	internal fluid	external fluid
Density	ρ	Kg/m^3	$\rho_0 = 1.0$	$\rho_1 = \text{ratio}^{-1}$
Dynamic viscosity	μ	Ns/m^2	10^{-4}	
Surface tension coefficient	σ	N/m	$\rho_1/1000$	
Initial ellipse axes	$(a,\, b)$	m	$(1.0,\, 0.5)$	

Table 1: Idealized parameters considered in the two-fluid flow test case.

In Figure 2 there is shown the impact of the density ratio on the spectrum of $\tilde{\mathsf{L}}$. Two things should be noted: firstly, the spectrum is presented normalized by the smallest (in magnitude) eigenvalue and ordered from smallest to largest, thus ranging, for all ratios, from 1 to the condition number, $\kappa(\tilde{\mathsf{L}}) = \lambda_{max}/\lambda_{min}$. Indeed: the higher the ratio, the much more ill-conditioned the system. Secondly, this effect becomes much more intense for finer meshes. Thus, we can assert that versions of $\tilde{\mathsf{L}}$ arising from large real case meshes and presenting important contrasts will be extremely ill-conditioned and, therefore, even though Equation 2 can only be solved by means of iterative solvers, its convergence will be excessively slow.

2.3 Preconditioning Poisson equation

It is clear that the solution by means of a direct application of CG (or any other iterative solver) is not reasonable, as large aspect ratios immediately translate into extremely ill-conditioned systems. There exists, however, a crucial tool to remedy this situation and achieve substantial improvements in the rate of convergence of the solvers: to precondition the linear system.



(a) 16×16 mesh

(b) 96×96 mesh

Figure 2: Normalized spectrum of $\tilde{\mathsf{L}}$ applied to the two-fluid test case for various density ratios and (homogeneous) meshes.

By preconditioning, we mean solving an equivalent system that results to be better conditioned. Hence, being solved in less iterations at the extra cost of having to set up the preconditioner, $K = K_1 K_2$, and to apply it at least once per iteration. Typically, this can be done in three different ways that arise from applying K (or its factors) to a different side of the coefficient matrix of a linear system $Ax = b$:

$$\begin{aligned} &\text{Left preconditioning:} && K^{-1}Ax = K^{-1}b \\ &\text{Right preconditioning:} && AK^{-1}y = b, \text{ where } Kx = y \\ &\text{Split preconditioning:} && K_1^{-1}AK_2^{-1}y = K_1^{-1}b, \text{ where } K_2x = y \end{aligned}$$

Obviously, the spectral properties of the preconditioned system, e.g. $K^{-1}Ax = K^{-1}b$, should be improved with respect to the unpreconditioned one, $Ax = b$. That is: $\kappa(K^{-1}A)$ should be closer to one than $\kappa(A)$, being its eigenvalues as clustered together as possible and, therefore, requiring the iterative solvers less iterations to converge. Moreover, if the solver being used requires the system to satisfy a certain condition, then the preconditioned system needs to satisfy it, too. From this constraint can be derived that K needs to be a symmetric positive-definite (SPD) matrix to be compatible with CG.

On the other side, setup and application overheads should be kept under control to not spoil the expected reduction in the total execution time of the preconditioned solvers. In this sense, it is crucial that the application of the preconditioner, which normally reduces to inverting K, is computationally affordable. This is the reason why diagonal and triangular preconditioners are so popular. As we already pointed, Equation 2 is particularly challenging, given that $\tilde{\mathsf{L}}$ is not only seriously ill-conditioned, but also variable. As a consequence, K will need to be updated according to the new $\tilde{\mathsf{L}}$ at each time iteration and, therefore, reasonably expensive setup overheads will no longer be affordable. The aim of this work is, then, to propose a set of variable preconditioners that can be updated at almost no extra cost while still performing equally well than other standard and widely used but unaffordable (in the variable case) preconditioners.

3 VARIABLE POISSON PRECONDITIONERS

The variable preconditioners that we propose arise as affordable, in the variable case, alternatives of already existing preconditioners. In spite of the fact that we are only going to analyze two "adaptive precontitioners", namely adaptive Jacobi and adaptive Incomplete Cholesky preconditioners, it is worth to mention that our approach gives rise to a whole new family whose shared main features have already been reviewed in the Introduction and whose derivation details will be discussed in Section 3.2.

3.1 Adaptive Jacobi preconditioner

Given the variable coefficients Poisson equation, Equation 2, defined by the variable discrete Laplace operator, $\tilde{\mathsf{L}}$, Jacobi preconditioner is defined as:

$$K_{\text{Jac}} \coloneqq \text{diag}(\tilde{\mathsf{L}}). \tag{5}$$

Despite being a rather simple preconditioner, in many cases its impact in the rate of convergence of the solvers may be quite notable. Moreover, its application is computationally very cheap and highly-parallelizable, as the product by the inverse of a diagonal matrix can be calculated trivially and does not involve communications between computing devices.

Even though the full matrix $\tilde{\mathsf{L}}$ is not required to extract its updated diagonal and build K_{Jac}, it must be said that its calculation may represent a significant overhead, specially in discretizations where Equation 1 is not verified and $\tilde{\mathsf{L}} \neq -\mathsf{M}\tilde{\Omega}^{-1}\mathsf{M}^t$, given that extra inter and intra-node communications may be necessary. In order to remedy this situation and arising from the observation that for large (and not so large) contrasts either in the coefficients or in the mesh sizes (R and Ω, respectively), the diagonal of $\tilde{\mathsf{L}}$ is dominated by $\tilde{\Omega}$ and the following approximation can be made:

$$\text{diag}(\tilde{\mathsf{L}}) \simeq \tilde{\Omega}^{-1} \tag{6}$$

leading to our adaptive Jacobi preconditioner:

$$\tilde{K}_{\text{Jac}} \coloneqq \tilde{\Omega}^{-1}. \tag{7}$$

whose action, as will be seen in Section 4, is very similar to that of K_{Jac} but eliminating the extra costs associated to calculating the updated diagonal of $\tilde{\mathsf{L}}$. This is done at a very small extra cost, namely one more diagonal matrix product, as $\tilde{K}_{\text{Jac}} = \mathsf{R}^{-1}\Omega^{-1}$ is formed by two factors while $K_{\text{Jac}} = \text{diag}(\tilde{\mathsf{L}})$ by one. In spite of that fact, one diagonal matrix product can be saved when either Ω or R are smooth, since that factor can be eliminated from $\tilde{K}_{\text{Jac}}$ without having any major impact on the spectrum of the preconditioned system. For instance, in the test case presented in Section 2.2 we can freely consider $\tilde{K}_{\text{Jac}} = \mathsf{R}^{-1}$, as the mesh is homogeneous and $\Omega = \Delta x \Delta y \mathbb{I}$. Additionally, it is worth to mention that the cost of obtaining the updated matrices Ω and R is considered negligible because in any simulation they will be required and updated in other parts of the code outside the solver.

3.2 Adaptive variants of existing preconditioners

Even though Jacobi preconditioner performs rather well considering its very low computational costs, in many cases it may result insufficient. For those cases there exist many other preconditioners, such as

those based on incomplete factorizations, multi-level approaches, polynomial series or sparse approximate inverses. A review of many well known and widely used preconditioners may be found in [15].

The problem of such more elaborated preconditioners is that they entail much higher setup costs which, as has already been discussed, are not affordable in the variable case because of having to rebuild K at each time iteration to adapt it to the new $\tilde{\mathsf{L}}$. Therefore, our proposal basically aims to drastically reduce the overhead linked to updating these more complex preconditioners while not losing much of their improved performances.

To do this, let us observe that, specially for very large discontinuities (both in R and Ω):

$$\sqrt{\tilde{\Omega}}\,\tilde{\mathsf{L}}\sqrt{\tilde{\Omega}} = -\sqrt{\tilde{\Omega}}\left(\mathsf{M}\tilde{\Omega}^{-1}\mathsf{M}^t\right)\sqrt{\tilde{\Omega}} \simeq -\mathsf{M}\mathsf{M}^t.$$

Hence, we propose building more complex preconditioners $K = FF^t$ based on the constant discrete Laplace operator and adapt them to $\tilde{\mathsf{L}}$ by combining them with $\tilde{K}_{\text{Jac}}$ in the following manner:

$$\tilde{K} \coloneqq \tilde{F}\tilde{F}^t, \text{ where } \tilde{F} \coloneqq \sqrt{\tilde{K}_{\text{Jac}}}F$$

thus having (via split preconditioning):

$$\tilde{F}^{-1}\tilde{\mathsf{L}}\tilde{F}^{-t} = F^{-1}\left(\sqrt{\tilde{K}_{\text{Jac}}}\right)^{-1}\tilde{\mathsf{L}}\left(\sqrt{\tilde{K}_{\text{Jac}}}\right)^{-1}F^{-t} \simeq F^{-1}\mathsf{L}F^{-t}$$

It should be noted that, as K is constant and will only need to be calculated once, then it still results affordable even if its setup costs are higher. Moreover, it will be based on $-\mathsf{M}\mathsf{M}^t$ only when both R and Ω are variable and present considerable contrasts. If either R or Ω are constant, then, according to what was discussed in the previous section, that constant factor will be ignored by $\tilde{K}_{\text{Jac}}$ and, in order to attain a better quality preconditioner, will need to be considered by K. Conversely, when either R or Ω are variable but very smooth (thus having a condition number close to 1 and not affecting noticeably the spectrum of $\tilde{\mathsf{L}}$), that factor could be ignored by both $\tilde{K}_{\text{Jac}}$ and K, hence not playing any role in $\tilde{F}$.

On the other side, the derivation of the general preconditioner $\tilde{K}$ has been made in a factored form to be consistent with the symmetry constraint imposed by CG. Nevertheless, if the variable coefficients Poisson equation is aimed to be solved using a less restrictive iterative method, then the adaptive version of a general and non-factored preconditioner K could be directly derived as:

$$\tilde{K} \coloneqq \begin{cases} \tilde{K}_{\text{Jac}}K, & \text{if left preconditioning} \\ K\tilde{K}_{\text{Jac}}, & \text{if right preconditioning} \end{cases}$$

a formulation that would, additionally, require less extra diagonal matrix products.

A particular example of such adaptive preconditioners could be based on the incomplete Cholesky preconditioner (cf. [15]) that, besides, is perfectly compatible with CG. As its name suggests, it is the result of performing an incomplete Cholesky factorization under many different possible strategies, such as imposing a fixed sparsity pattern or applying a certain dropping criterion on the nonzeros. Applying such preconditioner to $\tilde{\mathsf{L}}$ would result into:

$$K_{\text{IC}} \coloneqq L_{\tilde{\mathsf{L}}}L_{\tilde{\mathsf{L}}}^t \tag{8}$$

being $L_{\tilde{\mathsf{L}}}$ and $L^t_{\tilde{\mathsf{L}}}$ the incomplete Cholesky factors of $\tilde{\mathsf{L}}$. Clearly, even though an important reduction in the number of iterations needed to solve Equation 2 should be expected, it would not compensate the overhead of rebuilding the incomplete Cholesky factors at each time iteration (apart from the overhead of inverting two triangular matrices at each iteration of the solver). Under these circumstances our adaptive incomplete Cholesky preconditioner would read:

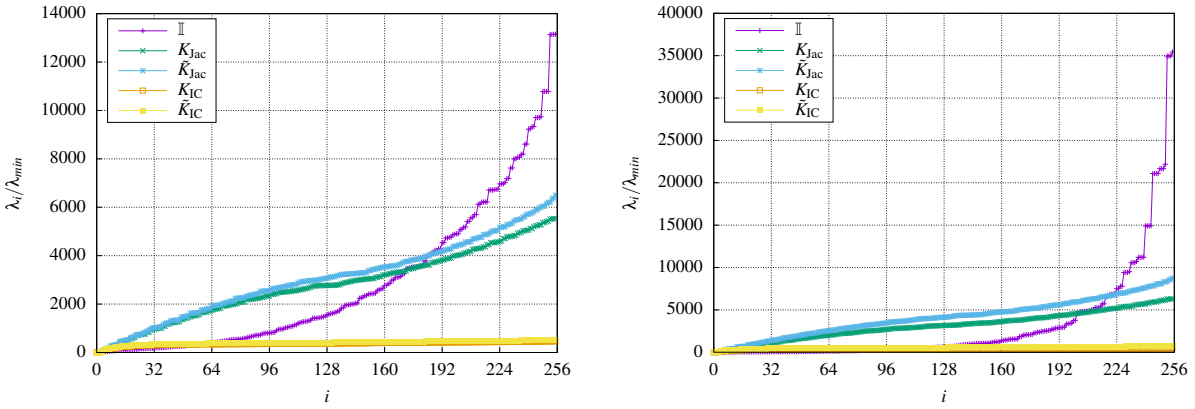
$$\tilde{K}_{\mathrm{IC}} \coloneqq \tilde{L}_{\mathsf{L}}\tilde{L}^t_{\mathsf{L}}, \text{ where } \tilde{L}_{\mathsf{L}} \coloneqq \sqrt{\tilde{K}_{\mathrm{Jac}}}L_{\mathsf{L}} \tag{9}$$

being L_{L} and L^t_{L} the incomplete Cholesky factors of L according to what was discussed above and eliminating the unaffordable overhead of rebuilding the incomplete factors at each time iteration.

4 NUMERICAL RESULTS

All the preconditioners presented in Section 3 have been implemented on MATLAB and tested for several ratios and meshes. Before analyzing the iterative solution of the preconditioned variable coefficients Poisson equation, let us review the spectral properties of the differently preconditioned operators, $K^{-1}\tilde{\mathsf{L}}$, in order to predict the behavior that should be exhibited by our proposals (according to theoretical results such as the convergence bound for CG given in Equation 4).

In Figure 3 there are shown the spectrums of different versions of $K^{-1}\tilde{\mathsf{L}}$ for ratios 10^2 and 10^6. Two immediate conclusions can be drawn from the plots: firstly, that the condition number of the unpreconditioned case, trivially designated as $K = \mathbb{I}$, is a lot higher than for the rest of cases. Indeed, as was already pointed with respect to Figure 2, with finer meshes this difference would become so large that the resulting plot would be badly scaled. Secondly, it is clear that the action of $\tilde{K}_{\mathrm{Jac}}$ and $\tilde{K}_{\mathrm{IC}}$ on $\tilde{\mathsf{L}}$ is very similar to that of K_{Jac} and K_{IC}, respectively, thus leading to comparable rates of convergence. Unsurprisingly, the more complex preconditioners based on incomplete Cholesky factorizations, K_{IC} and $\tilde{K}_{\mathrm{IC}}$, succeed on further clustering together the eigenvalues of $\tilde{\mathsf{L}}$.



(a) ratio = 10^2 (b) ratio = 10^6

Figure 3: Normalized spectrum of $K^{-1}\tilde{\mathsf{L}}$ for $K \in \left\{\mathbb{I}, K_{\mathrm{Jac}}, \tilde{K}_{\mathrm{Jac}}, K_{\mathrm{IC}}, \tilde{K}_{\mathrm{IC}}\right\}$ and applied to the two-fluid test case. Results obtained on a 16×16 homogeneous mesh.

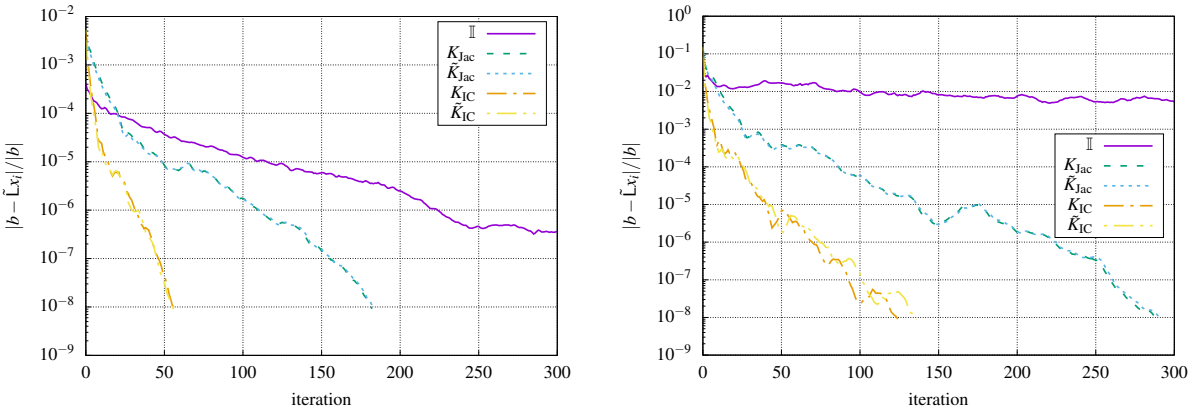
Finally, we have solved the same version of Equation 2 applied to the evolved two-fluid test case on a

96×96 homogeneous mesh using either of the preconditioners discussed so far. The different number of iterations required to converge (up to a relative tolerance of 1.0e−8) are presented in Table 1 for different ratios. The first column, $\mathbb{I}$, clearly shows how crucial preconditioning $\tilde{\mathsf{L}}$ results, specially for higher density ratios and finer meshes. Moreover, and as we had already predicted, it is clear that $\tilde{K}_{\text{Jac}}$ constitutes an excellent alternative to K_{Jac} when applied to $\tilde{\mathsf{L}}$ and for all the ratios considered, not necessarily extreme. In addition to $\tilde{K}_{\text{Jac}}$, $\tilde{K}_{\text{IC}}$ succeeds on further reducing the number of iterations with respect to $\tilde{K}_{\text{Jac}}$ and K_{Jac}, although it must be said that its discrepancy with respect to K_{IC} is slightly higher than that of $\tilde{K}_{\text{Jac}}$ with respect to K_{Jac}.

In any case, from the results of Table 1 and the convergence plots of Figure 4 we can clearly state that $\tilde{K}_{\text{Jac}}$, despite being a rather simple preconditioner and exactly as K_{Jac}, successfully preconditions high-ratio Poisson equation, while being particularly well-suited for the variable case. On the other side, $\tilde{K}_{\text{IC}}$ manages to further improve them both and makes available more complex preconditioners that are generally unaffordable in the variable case given their excessive update costs.

ratio	$\mathbb{I}$	K_{Jac}	$\tilde{K}_{\text{Jac}}$	K_{IC}	$\tilde{K}_{\text{IC}}$
1	152	151	152	46	46
10^2	611	182	183	56	56
10^4	5774	190	188	97	103
10^6	61731	288	291	124	136

Table 2: Number of iterations required by preconditioned CG to solve Equation 2 applied to the two-fluid test case for various preconditioners and ratios on a 96×96 homogeneous mesh with convergence criterion: $|b - \tilde{\mathsf{L}}x_i|/|b| < 1.0\text{e}{-8}$.



(a) ratio = 10^2 (b) ratio = 10^6

Figure 4: Evolution of the normalized residual, $|b - \tilde{\mathsf{L}}x_i|/|b|$, in the solution of Equation 2 applied to the two-fluid test case for various preconditioners and ratios on a 96×96 homogeneous mesh.

5 CONCLUDING REMARKS

As a conclusion, let us recall that a family of new "adaptive" preconditioners specially designed to suit variable versions of the Poisson equation with large contrasts in the coefficients has been presented.

On the one side, its simplest proposal arises as an alternative to the well-known Jacobi preconditioner, while, on the other, general rules to derive adaptive variants of more complex (and already existing but not affordable in the variable case) preconditioners are given. Among the characteristics of our different proposals, it is worth to mention that, even if they are variable and effectively adapted at each time iteration to the new linear system, they still entail a very low computational cost: their update stage is rather negligible, while the overhead related to their application comprises, at most, two diagonal matrix products that, in turn, are a very economical and highly parallelizable operation. Moreover, they do not require the full variable Laplace operator to be explicitly built and updated, which in highly parallel executions could represent an unacceptable overhead.

The numerical experiments conducted confirm that our proposals, namely adaptive Jacobi and incomplete Cholesky preconditioners, present very similar rates of convergence with respect to their analogues, thus leading to very important reductions in the number of iterations, specially for higher ratios and finer meshes, at a much lower computational cost.

As future lines of work, we plan to implement our adaptive preconditioners in real simulation codes to quantify how their negligible update costs actually translate into an important decrease of the total execution time of the simulations. More concretely, we aim to implement adaptive Jacobi, incomplete Cholesky and other adaptive variants of existing preconditioners in combination with HPC^2, our in-house algebraic library for heterogeneous computing (cf. [17]), and analyze their performance and scalability. Finally, we also plan to study the impact on the quality of our preconditioners of using different face-to-cell interpolators or of working with variable and highly-stretched meshes. Further possible lines of work could include the combination of our proposals with deflation techniques and multi-level approaches.

ACKNOWLEDGMENTS

Àdel Alsalti-Baldellou, F. Xavier Trias and Assensi Oliva have been financially supported by a competitive R+D project (ENE2017-88697-R) by the Spanish Research Agency. Àdel Alsalti-Baldellou is also supported by predoctoral grants DIN2018-010061 and 2019-DI-90, given by, respectively, the Spanish Ministry of Science, Innovation and Universities (MICINN) and the Catalan Agency for Management of University and Research Grants (AGAUR).

REFERENCES

[1] M. S. Dodd and A. Ferrante, "A fast pressure-correction method for incompressible two-fluid flows," *Journal of Computational Physics*, vol. 273, pp. 416–434, 2014.

[2] D. Pasetto, M. Ferronato, and M. Putti, "A reduced order model-based preconditioner for the efficient solution of transient diffusion equations," *International Journal for Numerical Methods in Engineering*, vol. 109, pp. 1159–1179, feb 2017.

[3] C. Vuik, A. Segal, and J. Meijerink, "An Efficient Preconditioned CG Method for the Solution of a Class of Layered Problems with Extreme Contrasts in the Coefficients," *Journal of Computational Physics*, vol. 152, pp. 385–403, jun 1999.

[4] R. K. Coomer and I. G. Graham, "Massively parallel methods for semiconductor device modelling," *Computing*, vol. 56, pp. 1–27, mar 1996.

[5] I. G. Graham and M. J. Hagger, "Unstructured Additive Schwarz–Conjugate Gradient Method for Elliptic Problems with Highly Discontinuous Coefficients," *SIAM Journal on Scientific Computing*, vol. 20, pp. 2041–2066, jan 1999.

[6] C. Vuik, A. Segal, L. El Yaakoubi, and E. Dufour, "A comparison of various deflation vectors applied to elliptic problems with discontinuous coefficients," *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 219–233, 2002.

[7] J. van der Linden, T. Jönsthövel, A. Lukyanov, and C. Vuik, "The parallel subdomain-levelset deflation method in reservoir simulation," *Journal of Computational Physics*, vol. 304, pp. 340–358, jan 2016.

[8] G. Diaz Cortes, C. Vuik, and J. Jansen, "On POD-based Deflation Vectors for DPCG applied to porous media problems," *Journal of Computational and Applied Mathematics*, vol. 330, pp. 193–213, mar 2018.

[9] B. Aksoylu, I. G. Graham, H. Klie, and R. Scheichl, "Towards a rigorously justified algebraic preconditioner for high-contrast diffusion problems," *Computing and Visualization in Science*, vol. 11, no. 4-6, pp. 319–331, 2008.

[10] A. J. Chorin, "Numerical solution of the Navier-Stokes equations," *Mathematics of Computation*, vol. 22, no. 104, pp. 745–745, 1968.

[11] R. Témam, "Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires (II)," *Archive for Rational Mechanics and Analysis*, vol. 33, pp. 377–385, jan 1969.

[12] N. Valle, F. Trias, and J. Castro, "An energy-preserving level set method for multiphase flows," *Journal of Computational Physics*, vol. 400, p. 108991, jan 2020.

[13] F. Trias, O. Lehmkuhl, A. Oliva, C. Pérez-Segarra, and R. Verstappen, "Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured grids," *Journal of Computational Physics*, vol. 258, pp. 246–267, feb 2014.

[14] R. W. Verstappen and A. E. Veldman, "Symmetry-preserving discretization of turbulent flow," *Journal of Computational Physics*, vol. 187, no. 1, pp. 343–368, 2003.

[15] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, vol. 56. Cambridge University Press, apr 2003.

[16] A. van der Sluis and H. A. van der Vorst, "The rate of convergence of Conjugate Gradients," *Numerische Mathematik*, vol. 48, pp. 543–560, sep 1986.

[17] X. Álvarez-Farré, A. Gorobets, and F. X. Trias, "A hierarchical parallel implementation for heterogeneous computing. Application to algebra-based CFD simulations on hybrid supercomputers," *Computers & Fluids*, vol. 214, p. 104768, jan 2021.