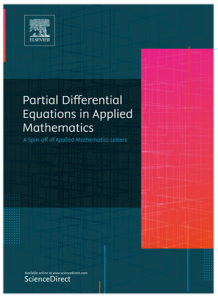
Journal Pre-proof



An iterative method for solving sparse linear algebraic systems with continuum solution dependent right-hand side for elliptic partial differential equations

Sudipta Lal Basu, Kirk M. Soodhalter, Breiffni Fitzgerald, Biswajit Basu

PII: S2666-8181(25)00163-9
DOI: https://doi.org/10.1016/j.padiff.2025.101236
Reference: PADIFF 101236

To appear in: *Partial Differential Equations in Applied Mathematics*
Received date : 18 March 2025
Accepted date : 2 June 2025

Please cite this article as: S.L. Basu, K.M. Soodhalter, B. Fitzgerald et al., An iterative method for solving sparse linear algebraic systems with continuum solution dependent right-hand side for elliptic partial differential equations, *Partial Differential Equations in Applied Mathematics* (2025), doi: https://doi.org/10.1016/j.padiff.2025.101236.

Highlights

An iterative method for solving sparse linear algebraic systems with continuum solution dependent right-hand side for elliptic partial differential equations

Sudipta Lal Basu, Kirk M. Soodhalter, Breiffni Fitzgerald, Biswajit Basu

- Systems governed by PDEs are discretized and solved using algorithms like BiCGStab.
- Certain real-world systems have unavailable continuum solution dependent right-hand side.
- A modified BiCGStab is proposed by coupling it with Decomposed Immersed Interface Method.
- The modified algorithm is validated against some benchmark problems.
- Application to real world problems and performance on multiple CPU cores is demonstrated.

An iterative method for solving sparse linear algebraic systems with continuum solution dependent right-hand side for elliptic partial differential equations

Sudipta Lal Basu[a], Kirk M. Soodhalter[b], Breiffni Fitzgerald[a], Biswajit Basu[a,*]

[a] *School of Engineering, Trinity College Dublin, Ireland*
[b] *School of Mathematics, Trinity College Dublin, Ireland*

Abstract

Krylov subspace iterative methods such as bi-conjugate gradients stabilized (BiCGStab) to approximately solve sparse linear algebraic systems are well known. However, there are certain instances in real-world engineering applications with underlying governing partial differential equation where the discretized right-hand side can only be exactly determined using the unavailable continuum solution. In such cases, an iterative method such as BiCGStab may not converge to a physically correct solution or may diverge completely. Such a method must be modified to accommodate inexact knowledge of the discrete right-hand side, using an updating scheme as the iteration proceeds. In this paper, we present such an updating strategy for physical problems governed by elliptic partial differential equations. This strategy must be performed in a numerically stable manner, which we also discuss. We present

*Corresponding author
Email addresses: basus@tcd.ie (Sudipta Lal Basu), ksoodha@maths.tcd.ie (Kirk M. Soodhalter), Breiffni.Fitzgerald@tcd.ie (Breiffni Fitzgerald), basub@tcd.ie (Biswajit Basu)

this as a modified BiCGStab iteration and investigate its effectiveness on both test problems, wherein it is shown to perform well and agrees with the analytical solutions, and on some more realistic problems arising in the study of Hele-Shaw flow, composite materials and power generation from wind farms.

Keywords: Iterative methods, Decomposed Immersed Interface Method, Interfacial problems, Fluid-structure interaction, Elliptic partial differential equations

2020 MSC: 65N06, 65F10, 65N22, 74F10

1. Introduction

Mathematical formulations and models for many real world engineering problems can be quite conveniently derived depending upon the parameters involved and the physical laws known. However, the same is not always true when deriving an analytical solution for these very same models. An example is the field of fluid dynamics where the mathematical models rely extensively on partial differential equations (PDEs) of forms for which the analytical solutions cannot be deduced straightaway. Thus, the solutions to complicated engineering models rely heavily on numerical techniques; in particular, iterative methods which generally must be used to treat the resulting large-scale linear systems in order to obtain appropriately accurate approximate solutions to these engineering models.

Optimal error-minimizing iterative methods like Conjugate Gradient(CG) and GMRES (generalized minimal residual method) are some of the common algorithms that are quite often used to solve for the aforementioned

systems of equations numerically with their choice dependent on whether or not the coefficient matrix is symmetric or non-symmetric. It has been demonstrated[1,2] that no general method exists to treat non-symmetric problems that both is a short-recurrence method and minimizes an error functional, such as CG. The discretization of the engineering problem treated in this paper results in linear systems that are non-symmetric and as such in this paper the focus is on algorithms dealing with these cases. GMRES is memory expensive and might not be suitable when a large domain is dealt with. Hence, the application of Biconjugate Gradient Stable (BiCGStab)[3] has been explored in the current context because of its inherent memory inexpensive approach. In the past, this was illustrated in a comparative study using GMRES, BiCGStab and Conjugate Gradient Squared (CGS) done to solve Navier-Stokes equation where the superior performance of BiCGStab with respect to computational resources was demonstrated[4]. Researchers have worked on interfacial flows using a local level set algorithm where they have used pre-conditioned form of BiCGStab to solve for elliptic PDE as well as for convection-diffusion step[5]. This clearly highlights the applicability of this algorithm to different forms of PDEs. Improvements to Krylov sub-space methods like CG and BiCGStab were proposed by using reduced-order models as solution predictors in order to accelerate the convergence[6]. Applications of this algorithm can also be found across works in multiple domains like hybrid plasma stability analysis[7], multi-component Cahn–Hilliard system on surfaces[8], fluid-structure interaction (FSI) in wind farms[9].

There has been much work in recent years in parallelizing iterative methods such as BiCGStab. For instance, for distributed computing systems

an approach called BiCGStab(2) was proposed which achieves efficiency by reducing global synchronization points to two from five and reducing communication time for inner products[10]. Preconditioned BiCGStab (PBiCGStab) using approximate inverse technique compatible with modern day GPUs have also been explored[11]. Other researchers have even proposed a parallel in time Stable Bi-Conjugate gradient algorithm which is applicable to para-real and initial value problems[12].

In this paper, we treat the paradoxical situation wherein we require knowledge of the continuum solution to construct the discrete right-hand side. None of the aforementioned efforts have been directed at cases wherein the construction of the discrete right hand side vector actually depends on knowledge of the continuum solution. In such cases, the user only has approximations to the discrete right-hand side which can be updated as the iteration proceeds. In this paper, an approach has been presented by proposing suitable modifications to BiCGStab to address this problem setup for elliptic PDEs. A detailed investigation of the modifications proposed has been carried out and the implications of the changes made when compared to the original BiCGStab has been discussed. Finally, some numerical examples from actual application problems have been presented to validate the proposed modifications along with examples of real world engineering problem to illustrate the application of the method. The proposed modifications do not restrict its ability of executing parallel computation across multiple CPU cores by implementing Message Passing Interface (MPI) which is evident from the simulations presented in this paper.

2. Defining the problem

We consider a sparse linear system of the form

$$\boldsymbol{A}\boldsymbol{\Phi} = \boldsymbol{b}. \tag{2.1}$$

where, $\boldsymbol{A}$ = coefficient matrix, $\boldsymbol{\Phi}$ = solution vector, $\boldsymbol{b}$ is the right-hand side. In this paper, we consider the situation wherein the right-hand side is split into two components, $\boldsymbol{b} = \boldsymbol{b}_c + \boldsymbol{b}_v$; the component $\boldsymbol{b}_c$ represents the part of the right-hand side known a priori. The other component $\boldsymbol{b}_v$ represents the part of the right-hand side which can be constructed exactly only with additional, unavailable information. For example, consider the case in which we construct a finite-difference discretization of a PDE. There are cases in which the boundary conditions and solution interact such that one would need the exact continuum solution Φ to exactly determine how the discrete boundary conditions manifest at certain nodes. If the problem has Dirichlet boundary conditions, this translates to elements of the right-hand side not being known exactly.

The family of BiCGStab methods operate by alternating between steps of a BiCG-type iteration and steps of a GMRES-type iteration. This confers on the method the advantages of a non-symmetric Lanczos-based short-recurrence method like BiCG with a more stabilized residual convergence pattern produced by a GMRES-type iteration, thereby reducing numerical issues caused by residual norms oscillating wildly between iterations. In this paper, we use a version of BiCGStab which alternates between one step of preconditioned BiCG and one step of preconditioned GMRES.

In the standard case wherein $\boldsymbol{b}$ is known exactly a priori, we simply set

the initial residual, $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{\Phi}_0$. However, since that is not the case, these conventional algorithms cannot be applied out-of-the-box and require modifications. Implicitly, there are a two sets of unknowns, the actual solution vector $\boldsymbol{\Phi}$ and the components of the right-hand side which are not known exactly, $\boldsymbol{b}_v := \boldsymbol{b}_v(\Phi)$, which denotes to be a function of the true continuum solution Φ. Since we will never have access to the true continuum solution Φ, we are only able to construct approximations of $\boldsymbol{b}_v(\Phi)$ using the approximate, discrete solution $\boldsymbol{\Phi}$. Thus, when given a discrete argument, $\boldsymbol{b}_v(\boldsymbol{\Phi})$ is taken to be a function mapping $\mathbb{R}^n \to \mathbb{R}^n$.

In this paper, bold characters denote matrices and vectors. Our iterative algorithm of choice for treating eq. (2.1) is a BiCGStab type short recurrence method[3]. This method is proposed for treating non-symmetric problems, but we argue in this paper that it has a valid use-case. The steps involved in BiCGStab algorithm are presented in algorithm 1 and the preconditioned form is given in algorithm 2. The approach presented in this paper proposes a modification to the BiCGStab iteration that simultaneously updates approximation, i.e., $\boldsymbol{\Phi}_{i-1} \to \boldsymbol{\Phi}_i$, as well as an approximation for $\boldsymbol{b}_v$ in a way that improves the stability of the iteration (increasing the attainable accuracy of the approximation $\boldsymbol{\Phi}_i$).

The assumption we make in this paper is that although we do not have access to $\boldsymbol{b}_v$, we have access to the function $\boldsymbol{b}_v(\boldsymbol{x})$ (via using the argument to update our knowledge of the discrete right-hand side at the boundary nodes), which returns $\boldsymbol{b}_v = \boldsymbol{b}_v(\Phi)$ when the true solution is given as the argument. Given an initial approximation $\boldsymbol{\Phi}_0$, we cannot compute the exact initial residual $\boldsymbol{r}_0$ because we do not know $\boldsymbol{b}_v$. Instead, we compute $\boldsymbol{r}_0 =$

Algorithm 1 Biconjugate Gradient Stable[3]

Assume initial trial value for vector $\boldsymbol{\Phi}$, say $\boldsymbol{\Phi}_0$.

$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{\Phi}_0$.

Choose an arbitrary vector $\boldsymbol{r}^*$ such that $< \boldsymbol{r}^*, \boldsymbol{r}_0 > \neq 0$, e.g., $\boldsymbol{r}^* \leftarrow \boldsymbol{r}_0$.

$\rho_0 = \alpha = \omega_0 = 1$.

$\boldsymbol{\nu}_0 = \boldsymbol{p}_0 = 0$.

for $j = 1, 2, 3, ...$ **do**

$\rho_j \leftarrow < \boldsymbol{r}^*, \boldsymbol{r}_{j-1} >$; $\beta \leftarrow (\rho_j / \rho_{j-1})(\alpha / \omega_{j-1})$

$\boldsymbol{p}_j \leftarrow \boldsymbol{r}_{j-1} + \beta(\boldsymbol{p}_{j-1} - \omega_{j-1}\boldsymbol{\nu}_{j-1})$

$\boldsymbol{\nu}_j \leftarrow \boldsymbol{A}\boldsymbol{p}_j$

$\alpha \leftarrow \rho_j / < \boldsymbol{r}^*, \boldsymbol{\nu}_j >$

$\boldsymbol{s} \leftarrow \boldsymbol{r}_{j-1} - \alpha\boldsymbol{\nu}_j$

$\boldsymbol{t} \leftarrow \boldsymbol{A}\boldsymbol{z}$

$\omega_j \leftarrow < \boldsymbol{t}, \boldsymbol{s} > / < \boldsymbol{t}, \boldsymbol{t} >$

$\boldsymbol{\Phi}_j \leftarrow \boldsymbol{\Phi}_{j-1} + \alpha\boldsymbol{p}_j + \omega_j\boldsymbol{z}$; if $\boldsymbol{\Phi}_j$ accurate enough then quit

$\boldsymbol{r}_j \leftarrow \boldsymbol{s} - \omega_j\boldsymbol{t}$

end for

Algorithm 2 Preconditioned BiConjugate Gradient Stable (PBiCGStab)[3]

Assume initial trial value for vector $\boldsymbol{\Phi}$, say $\boldsymbol{\Phi}_0$.

$\boldsymbol{r}_0 \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{\Phi}_0$.

Choose an arbitrary vector $\boldsymbol{r^*}$ such that $< \boldsymbol{r^*}, \boldsymbol{r}_0 > \neq 0$, e.g., $\boldsymbol{r^*} \leftarrow \boldsymbol{r}_0$

$\rho_0 = \alpha = \omega_0 = 1$

$\boldsymbol{\nu}_0 = \boldsymbol{p}_0 = 0$

for j = 1, 2, 3, ... **do**

$\rho_j \leftarrow < \boldsymbol{r^*}, \boldsymbol{r}_{j-1} >$; $\beta = (\rho_j/\rho_{j-1})(\alpha/\omega_{j-1})$

$\boldsymbol{p}_j \leftarrow \boldsymbol{r}_{j-1} + \beta(\boldsymbol{p}_{j-1} - \omega_{j-1}\boldsymbol{\nu}_{j-1})$

$\boldsymbol{y} \leftarrow \boldsymbol{M}^{-1}\boldsymbol{p}_j$

$\boldsymbol{\nu}_j \leftarrow \boldsymbol{A}\boldsymbol{y}$

$\alpha \leftarrow \rho_j / < \boldsymbol{r^*}, \boldsymbol{\nu}_j >$

$\boldsymbol{s} \leftarrow \boldsymbol{r}_{j-1} - \alpha\boldsymbol{\nu}_i$

$\boldsymbol{z} \leftarrow \boldsymbol{M}^{-1}\boldsymbol{s}$

$\boldsymbol{t} \leftarrow \boldsymbol{A}\boldsymbol{z}$

$\omega_j \leftarrow < \boldsymbol{M}^{-1}\boldsymbol{t}, \boldsymbol{M}^{-1}\boldsymbol{s} > / < \boldsymbol{M}^{-1}\boldsymbol{t}, \boldsymbol{M}^{-1}\boldsymbol{t} >$

$\boldsymbol{\Phi}_j \leftarrow \boldsymbol{\Phi}_{j-1} + \alpha\boldsymbol{y} + \omega_j\boldsymbol{z}$; if $\boldsymbol{\Phi}_j$ accurate enough then quit

$\boldsymbol{r}_j \leftarrow \boldsymbol{s} - \omega_j\boldsymbol{t}$

end for

$\boldsymbol{b}_c + \boldsymbol{b}_v(\boldsymbol{\Phi}_0) - \boldsymbol{A}\boldsymbol{\Phi}_0$. Following suggestions in, e.g., [13], the arbitrary vector $\boldsymbol{r}^*$ is assigned the same value as $\boldsymbol{r}_0$. At the end of iteration j, the updated solution vector and the residuals are computed by executing the step

$$\boldsymbol{\Phi}_j \leftarrow \boldsymbol{\Phi}_{j-1} + \alpha \boldsymbol{p}_j + \omega_j \boldsymbol{z};\ \boldsymbol{r}_j \leftarrow \boldsymbol{s} - \omega_j \boldsymbol{t} \text{ (BiCGStab)}$$

or

$$\boldsymbol{\Phi}_j \leftarrow \boldsymbol{\Phi}_{j-1} + \alpha \boldsymbol{y} + \omega_j \boldsymbol{z};\ \boldsymbol{r}_j \leftarrow \boldsymbol{s} - \omega_j \boldsymbol{t} \text{ (PBiCGStab)}$$

respectively. Assuming that for the choice $\boldsymbol{r}^* \leftarrow \boldsymbol{r}_0$, this iteration converges, the solution thus obtained would be the solution of the linear system

$$\boldsymbol{A}\boldsymbol{\Phi} = \boldsymbol{b}_c + \boldsymbol{b}_v(\boldsymbol{\Phi}_0)$$

instead of

$$\boldsymbol{A}\boldsymbol{\Phi} = \boldsymbol{b}.$$

If the construction of $\boldsymbol{b}$ were independent of $\boldsymbol{\Phi}$, then we would have

$$\boldsymbol{r}_j = \boldsymbol{s} - \omega_j \boldsymbol{t} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{\Phi}_j.$$

Instead, with the approximation of $\boldsymbol{b}_v$, it follows that

$$\boldsymbol{r}_j = \boldsymbol{s} - \omega_j \boldsymbol{t} = \boldsymbol{b}_c + \boldsymbol{b}_v(\boldsymbol{\Phi}_0) - \boldsymbol{A}\boldsymbol{\Phi}_j.$$

We propose a modification of the BiCGStab iteration wherein the approximation of $\boldsymbol{b}_v$ is updated by evaluating $\boldsymbol{b}_v(\cdot)$ using the new approximation $\boldsymbol{\Phi}_j$. In principle, this would lead to the update

$$\boldsymbol{r}_j = \boldsymbol{s} - \omega_j \boldsymbol{t} = \boldsymbol{b}_c + \boldsymbol{b}_{v_j}(\boldsymbol{\Phi}_j) - \boldsymbol{A}\boldsymbol{\Phi}_j$$

This means that the residual at the end of each iteration should be updated such that the updated approximation of the right-hand side is reflected in

the residual, $\boldsymbol{r}_j$ at the end of iteration, j. Thus, we update the residual $\boldsymbol{r}_j$ accordingly. Let $\boldsymbol{C}_j$ be the jth correction vector, used to effect the current residual update, defined as

$$\boldsymbol{C}_j = -\boldsymbol{b}_{v_{j-1}}(\boldsymbol{\Phi}_{j-1}) + \boldsymbol{b}_{v_j}(\boldsymbol{\Phi}_j). \tag{2.2}$$

It then follows that $\boldsymbol{r}_j$ is corrected via

$$\boldsymbol{r}_j = \boldsymbol{s} - \omega_j \boldsymbol{t} + \boldsymbol{C}_j. \tag{2.3}$$

The BiCGStab algorithm allows for $\boldsymbol{r}^*$ to be chosen arbitrarily, independent of $\boldsymbol{r}_0$. However, as it has been observed by multiple authors (based on heuristic observations)[13], the choice of $\boldsymbol{r}^* = \boldsymbol{r}_0$ is often the most effective one. Indeed, in simplified versions of the application problem we treat in this paper (cf. section 3), it was difficult to find examples for which BiCGStab would converge for other choices of $\boldsymbol{r}^*$. As such, as we refine our estimate of $\boldsymbol{b}_v$ using $\boldsymbol{\Phi}_j$, it is essential that $\boldsymbol{r}^*$ be updated, which can be achieved via assigning the index $\boldsymbol{r}_j^*$, setting $\boldsymbol{r}_0^* = \boldsymbol{b}_c + \boldsymbol{b}_v(\boldsymbol{\Phi}_0) - \boldsymbol{A}\boldsymbol{\Phi}_0$, and updating

$$\boldsymbol{r}_j^* = \boldsymbol{r}_{j-1}^* + \boldsymbol{C}_j. \tag{2.4}$$

2.1. The use of BiCGStab in this setting

A modified right PBiCGStab is implemented to carry out the iterations. If the internal discontinuities/porous boundaries were not present, with this algorithm, the residual, $\boldsymbol{r_{j-1}} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{\Phi}_{j-1}$ would have been updated at the end of each iteration step with the updated vector $\boldsymbol{\Phi}_j$. However, to incorporate correction to the irregular nodes, the residual is modified to $\boldsymbol{r} = \boldsymbol{b}_c - \boldsymbol{A}\boldsymbol{\Phi} - \boldsymbol{b}_v(\boldsymbol{\Phi}_{j-1}) + \boldsymbol{b}_v(\boldsymbol{\Phi}_j)$ instead.

Taking a step back, depending on the function $\boldsymbol{b}_v(\cdot)$, this is potentially a non-linear problem. Given $\boldsymbol{A}$ and right-hand side component $\boldsymbol{b}_c$, fixed, we can define the set of feasible solutions

$$\mathcal{S}_{\boldsymbol{A},\boldsymbol{b}_c} = \{\boldsymbol{\Phi} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{\Phi} = \boldsymbol{b}_c + \boldsymbol{b}_v(\boldsymbol{\Phi})\}.$$

Depending on the nature of the function $\boldsymbol{b}_v(\cdot)$, $\mathcal{S}_{\boldsymbol{A},\boldsymbol{b}_c}$ may be a single point, a discrete family of points, or something more complicated. Indeed, for the PDE example in cf. section 3, the PDE is well-posed and should have only one solution. However, for the discrete linear system, $\boldsymbol{b}_v(\cdot)$ is defined by a process of constructing interpolation polynomials along immersed boundary normal directions, which could potentially be nonlinear. We frame this discussion in the more general case, to account for this possibility.

We make the well-known observation that algorithm 2 can conceptually be simplified in terms of the action of some projectors, which we encapsulate as algorithm 3, where we define the projectors

$$\begin{aligned}
\boldsymbol{Q}_1^{(i)}(\boldsymbol{z}) &= \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{z}(\boldsymbol{r}^{\boldsymbol{*}T}_i\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{z})^{-1}\boldsymbol{r}^{\boldsymbol{*}T}_i \\
\boldsymbol{P}_1^{(i)}(\boldsymbol{z}) &= \boldsymbol{M}^{-1}\boldsymbol{z}(\boldsymbol{r}^{\boldsymbol{*}T}_i\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{z})^{-1}\boldsymbol{r}^{\boldsymbol{*}T}_i\boldsymbol{A} \\
\boldsymbol{Q}_2^{(i)}(\boldsymbol{z}) &= \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{z}((\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{z})^T\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{z})^{-1}(\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{z})^T \\
\boldsymbol{P}_2^{(i)}(\boldsymbol{z}) &= \boldsymbol{M}^{-1}\boldsymbol{z}((\boldsymbol{M}^{-1}\boldsymbol{z})^T(\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{M}^{-1}\boldsymbol{z})^{-1}(\boldsymbol{M}^{-1}\boldsymbol{z})^T(\boldsymbol{A}^T\boldsymbol{A}),
\end{aligned}$$

wherein we note that in algorithm 3, $\boldsymbol{I} - \boldsymbol{Q}_1^{(i)}(\boldsymbol{r}_j)$ projects $\boldsymbol{r}_j$ onto $\{\boldsymbol{r}^{\boldsymbol{*}}{}_i\}^{\perp}$ along $\{\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{r}_j\}$ and $\boldsymbol{I} - \boldsymbol{P}_1^{(i)}(\boldsymbol{r}_j)$ projects $\boldsymbol{\eta}_j = \boldsymbol{A}^{-1}\boldsymbol{r}_j$ onto $\left\{\boldsymbol{A}^T\boldsymbol{r}^{\boldsymbol{*}}{}_i\right\}^{\perp}$ along $\{\boldsymbol{M}^{-1}\boldsymbol{r}_j\}$, while $\boldsymbol{I}-\boldsymbol{Q}_2^{(i)}(\boldsymbol{r}_j)$ orthogonally projects $(\boldsymbol{I}-\boldsymbol{Q}_1^{(i)}(\boldsymbol{r}_j))\boldsymbol{r}_j$ onto $\{\boldsymbol{A}\boldsymbol{M}^{-1}(\boldsymbol{I} - \boldsymbol{Q}_1^{(i)}(\boldsymbol{r}_j))\boldsymbol{r}_j\Big\}^{\perp}$ and $\boldsymbol{I}-\boldsymbol{P}_1^{(i)}(\boldsymbol{r}_j)$ $\boldsymbol{A}^T\boldsymbol{A}$-orthogonally projects $(\boldsymbol{I}-\boldsymbol{P}_1^{(i)}(\boldsymbol{r}_j))\boldsymbol{\eta}_j)$ onto $\Big\{\boldsymbol{M}^{-1}(\boldsymbol{I} - \boldsymbol{Q}_1^{(i)}(\boldsymbol{r}_j))\boldsymbol{r}_j\Big\}^{\perp}$.

Algorithm 3 Conceptual PBiCGStab with projectors and right-hand side correction[3]

Assume initial approximate solution for vector $\boldsymbol{\Phi}_0$.

if $\boldsymbol{b}_v(\boldsymbol{\Phi}_0) = 0$ **then** flag = True

else flag = False

end if

Set $\boldsymbol{r}_0 = \boldsymbol{b}_c + \boldsymbol{b}_v(\boldsymbol{\Phi}_0) - \boldsymbol{A}\boldsymbol{\Phi}_0$.

Set $\boldsymbol{r}_0^* = \boldsymbol{r}_0$

for $j = 1, 2, 3, \ldots$ **do**

Set $\boldsymbol{r}_j = \boldsymbol{r}_{j-1}$, $\boldsymbol{\Phi}_j^{(i)} = \boldsymbol{\Phi}_{j-1}^{(i)}$, and $\boldsymbol{\eta}_j = \boldsymbol{A}^{-1}\boldsymbol{r}_j$.

$\boldsymbol{r}_j \leftarrow (\boldsymbol{I} - \boldsymbol{Q}_1^{(i)}(\boldsymbol{r}_j))\boldsymbol{r}_j$ and $\boldsymbol{\Phi}_j \leftarrow \boldsymbol{\Phi}_j + \boldsymbol{P}_1^{(i)}(\boldsymbol{r}_j)\boldsymbol{\eta}_j$.

$\boldsymbol{r}_j \leftarrow (\boldsymbol{I} - \boldsymbol{Q}_2^{(i)}(\boldsymbol{r}_j))\boldsymbol{r}_j$ and $\boldsymbol{\Phi}_j^{(i)} \leftarrow \boldsymbol{\Phi}_j^{(i)} + \boldsymbol{P}_2^{(i)}(\boldsymbol{r}_j)\boldsymbol{\eta}_j$.

if $-\boldsymbol{b}_v(\boldsymbol{\Phi}_{j-1}) + \boldsymbol{b}_v(\boldsymbol{\Phi}_j) \neq 0$ or flag=True **then**

$\boldsymbol{r}_j \leftarrow \boldsymbol{r}_j - \boldsymbol{b}_v(\boldsymbol{\Phi}_{j-1}) + \boldsymbol{b}_v(\boldsymbol{\Phi}_j)$

Set $\boldsymbol{r}_j^* = \boldsymbol{r}_{j-1}^* - \boldsymbol{b}_v(\boldsymbol{\Phi}_{j-1}) + \boldsymbol{b}_v(\boldsymbol{\Phi}_j)$

end if

if flag=True and $\boldsymbol{b}_v(\boldsymbol{\Phi}_{j-1}) = \boldsymbol{0}$ and $\boldsymbol{b}_v(\boldsymbol{\Phi}_j) = \boldsymbol{0}$ **then** continue

else Set flag=False

end if

end for

Note that were this algorithm executed with the knowledge of the true right-hand side $\boldsymbol{b}$, we would simply be discussing residual and error projection, which is a well-understood way of describing a method such as BiCGStab. Instead, BiCGStab becomes a nonlinear iterative scheme when applied to problems of the form (2.1) with b_ν being an unknown function of the solution.

We provide some analysis of the behaviour of the norm of the residual in section 4, which illustrates the importance of updating this arbitrary vector, $\boldsymbol{r}^*$. An important observation in this context is the behaviour of $\boldsymbol{C}_j$. As the method converges, it follows that $\boldsymbol{b}_v(\boldsymbol{\Phi_j}) \approx \boldsymbol{b}_v(\boldsymbol{\Phi_{j-1}})$, as long as we assume $\boldsymbol{b}_v(\cdot)$ is sufficiently smooth. This implies that $\|\boldsymbol{C}_j\| = \epsilon \ll 1$. As ϵ approaches machine precision, this can lead to the introduction of numerical errors due to phenomena such as cancellation. In particular, if left unchecked, this will start introducing an unwanted numerical noise into the iterations at the stage when $\|\boldsymbol{r}_j\| \sim \epsilon$, which can cause the residual to oscillate even when it has reached close to the numerical solution. One way to circumvent this is to stop correcting the residuals any further. In certain scenarios, if $\boldsymbol{bv}(\boldsymbol{\Phi_0}) = 0$, then $\boldsymbol{C}_j$ automatically becomes 0 which may not be correct. If this happens, the modified algorithm in itself will not be activated and the behaviour will be the same as the original BiCGStab. The if conditions ensure that these two situations in the algorithm 3 are appropriately addressed. A detailed numerical example portraying the behaviour of $||\boldsymbol{r}||_2$ has been presented in section 4.

3. Setting up the model

Problems with variable right hand side vector are not uncommon in interfacial (FSI, composite materials, etc.) problems. The study taken up and the examples presented in this present work focusses on the application of Decomposed Immersed Interface Method (DIIM)[14] to interfacial problems wherein the right hand side vector is dependent on the solution vector. Though this finite difference based method is originally proposed for elliptic partial differential equations (PDEs), it is claimed that it could be extended to other PDEs as well. However, for the current paper, application to other PDEs or problems with irregular grids is not investigated.

In this paper, elliptic PDEs of the form

$$\frac{\partial}{\partial X_1}\left(\beta\frac{\partial\phi}{\partial X_1}\right)+\frac{\partial}{\partial X_3}\left(\beta\frac{\partial\phi}{\partial X_3}\right)=-b;(X_1,X_3)\in\Omega \tag{3.1}$$

is taken up for study where, ϕ is the dependent variable, $\beta\partial\phi/\partial X_i=\beta(X_1,X_3)$ $\partial\phi/\partial X_i; i\in\{1,3\}$ is the flux across an interface, Γ inside a domain (refer fig. 1), Ω, $b=b(X_1,X_3)$ is the source term and the two dimensional (2D) domain, Ω is defined by $[c,d]\times[e,f]$; c and d are lower and upper bounds of Ω along X_1 axis and e and f are lower and upper bounds of Ω along X_3 axis . The source term, $-b(X_1,X_3)$ and the flux term, $\beta\partial\phi/\partial X_i$ are assumed to be continuous and smooth on each subdomain enclosed by a boundary. For grid nodes spaced equally, the node-to-node distance is defined by $\triangle X_1=(d-c)/(M-1), \triangle X_3=(f-e)/(N-1)$, where M,N are the number of grid points along axes X_1 and X_3 respectively. This gives the node coordinates as

$$X_1=c+i\triangle X_1;\{i:0\leq i<M, i\in\mathbb{W}\}$$

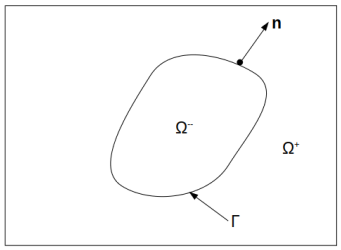


Fig. 1: Irregular interface Γ dividing overall domain Ω into Ω^+ and Ω^-

$$X_3 = e + j \triangle X_3; \{j : 0 \leq j < N, j \in \mathbb{W}\} \tag{3.2}$$

The application of DIIM was demonstrated for smaller fluid domains with a single discontinuity and with simple red-black Gauss-Seidel iterations. However, the method is not explored in greater detail for a larger physical domain in the past. For a large linear system Gauss-Seidel is not efficient which is where iterative algorithms like BiCGStab come into play. Hence, in this paper DIIM applied to different scenarios is used in order to demonstrate the performance of the modifications proposed to BiCGStab.

3.1. A brief introduction to DIIM

The method is based on a prior knowledge of two main conditions i.e. the change in value of ϕ and the change in the normal flux across Γ (refer fig. 1). These changes, often referred to as 'jumps' are defined by

$$\begin{aligned}[\phi] &= \lim_{(X_1,X_3)\to\Gamma^+} \phi(X_1, X_3) - \lim_{(X_1,X_3)\to\Gamma^-} \phi(X_1, X_3) \\ &= w(X_1, X_3)\end{aligned} \tag{3.3}$$

$$
\begin{aligned}
[\beta\phi_n] &= \lim_{(X_1,X_3)\to\Gamma^+} \beta(X_1,X_3)\phi_n(X_1,X_3) - \lim_{(X_1,X_3)\to\Gamma^-} \beta(X_1,X_3)\phi_n(X_1,X_3) \\
&= v(X_1,X_3)
\end{aligned} \tag{3.4}
$$

where, $[\phi] = w(X_1, X_3)$ is jump in value of ϕ, $[\beta\phi_n] = v(X_1, X_3)$ is jump in value of derivative in direction $\boldsymbol{n}$. With reference to fig. 1, it can be seen that Γ divides the overall domain, Ω into sub-domains Ω^+ and Ω^- which refer to the regions outside and inside respectively with respect to the region enclosed by Γ.

In general, IIM strategies classify the nodes inside a domain as regular and irregular. A node is classified as regular if all its dependent nodes are on the same side of Γ. For instance, with respect to the standard 5-point stencil for elliptic PDEs, if a node in Ω^+ lying on the left side of Ω^- and very near to Γ is considered, then its right side node will naturally lie inside Ω^-. As such, this node will be classified as irregular. Similarly, if hyperbolic or parabolic PDEs are looked at, which do not depend on 5-point stencil or even for elliptic PDEs with higher order differencing, a node, N_i will be classified as regular only if the finite difference equation (FDE) framed for N_i is dependent on a set of surrounding nodes, S such that $\{N_i, S\} \in \Omega^+$ or $\{N_i, S\} \in \Omega^-$.

For eq. (3.1), the FDE using the standard 5 point stencil can be written as

$$
\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\triangle X_1^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\triangle X_3^2} = - b_{i,j} - C_{i,j}
$$

$$\Rightarrow b_{i,j} + C_{i,j} - s\phi_{i,j} - p(\phi_{i+1,j} + \phi_{i-1,j}) - r(\phi_{i,j+1} + \phi_{i,j-1}) = 0. \tag{3.5}$$

where, $p = -\frac{1}{\triangle X_1^2}$, $r = -\frac{1}{\triangle X_3^2}$, $s = -2(p + r)$. It can be observed that there is an additional term, $C_{i,j}$ in eq. (3.5) called the correction term which DIIM introduces for the irregular nodes only at every stage of an iteration and is dependent on $w(X_1, X_3)$ and $v(X_1, X_3)$. It is evident that this value is dependent on the instantaneous value of $\phi_{i,j}$. The detailed approach to calculate $C_{i,j}$ using Taylor series approximation and second order Lagrange polynomial is not discussed here (refer[14] for details). However, the python code developed by the authors does take into account all these factors to compute $C_{i,j}$ for the numerical examples presented in this paper. It is to be noted that the approximation of the correction term, $C_{i,j}$ needs to be under-relaxed in order to ensure the numerical stability. This is done by introducing an under-relaxation parameter, α such that $\{\alpha : 0 < \alpha < 1, \alpha \in \mathbb{R}\}$.

$$C_{i,j}^{l} = C_{i,j}^{l-1} - \alpha(C_{i,j}^{l-1} - C_{i,j}^{new}) \tag{3.6}$$

where, l and $l-1$ denote the iteration number. Theoretically the order of algebraic operations might seem to be fine. However, it has significant numerical implication as it might introduce larger floating point errors. As such in the current work, the algebraic operations are performed instead as per eq. (3.7).

$$C_{i,j}^{l} = (1 - \alpha)C_{i,j}^{l-1} + \alpha C_{i,j}^{new} \tag{3.7}$$

Usually $\{C_{i,j} : C_{i,j} \in \mathbb{R}\}$. Theoretically as the iterations proceed and the solution converges, $C_{i,j}^{l-1} = C_{i,j}^{new}$ and this eventually results in $C_{i,j}^{l} = C_{i,j}^{l-1}$.

However, numerically there is always a loss in floating point at every step of iteration. Essentially this would imply

$$\begin{aligned} C_{i,j}^{l-1} \approx C_{i,j}^{new} &\Rightarrow |C_{i,j}^{l-1} - C_{i,j}^{new}| = \epsilon' \ (0 < \epsilon' < 1) \\ &\Rightarrow |\alpha(C_{i,j}^{l-1} - C_{i,j}^{new})| \approx \alpha\epsilon' << 1 \end{aligned} \tag{3.8}$$

This factor $\alpha\epsilon'$ starts introducing a numerical noise in eq. (3.6) as $C_{i,j}^{l-1} \not\sim \alpha\epsilon'$. On the contrary, if eq. (3.7) is considered, both the terms that are getting added are of the same order of magnitude i.e. $(1-\alpha)C_{i,j}^{l-1} \sim \alpha C_{i,j}^{new}$ and as such no numerical noise is generated by the operation itself.

The simulations presented in this paper use α within the range $0.01-0.05$. As with most problems governed by PDEs for which numerical simulations are carried out using approximate FDEs, the value of α is problem dependent and has to be chosen heuristically depending on the particular problem. From the brief description of DIIM presented above, it is evident that the term, $C_{i,j}$ makes the right hand vector $\boldsymbol{b}$ dependent on solution vector, $\boldsymbol{\Phi}$ (vector of dependent variable, ϕ).

3.2. Setting up the pre-conditioner

In this section, we describe how to set up a typical pre-conditioner for the examples presented in this paper is discussed. In order to ensure parallelism for faster processing of a large linear system, red-black coloring scheme is implemented using the partitioning scheme presented in fig. 2. Accordingly the matrix structure of $\boldsymbol{A}$ is split into blocks of smaller matrices as shown in eq. (3.9).

$$\boldsymbol{A\Phi} = \begin{pmatrix} \boldsymbol{D}_{re} & \boldsymbol{F} \\ \boldsymbol{E} & \boldsymbol{D}_{bl} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Phi}_{re} \\ \boldsymbol{\Phi}_{bl} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b'}_{re} \\ \boldsymbol{b'}_{bl} \end{pmatrix} \tag{3.9}$$

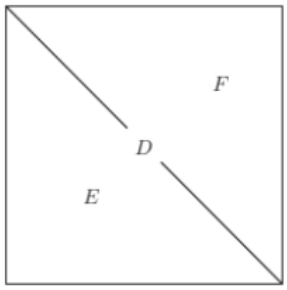


Fig. 2: Initial partitioning of **A**

where,

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{D}_{re} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{bl}, \end{pmatrix}, \boldsymbol{E} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{E} & \boldsymbol{0} \end{pmatrix},$$

$$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{F} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}, \boldsymbol{0} = \text{zero matrix}$$

and suffixes re and bl denote matrices corresponding to red and black nodes. It is to be noted that matrices $\boldsymbol{E}$ and $\boldsymbol{F}$ are padded with zero matrices for compatibility of the dimensions such that $\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{E} + \boldsymbol{F}$. A good pre-conditioner is required to make the solution of the linear system to converge faster. However, designing an optimal pre-conditioner is beyond the scope of this paper and as such for the purpose of the current work, standard Symmetric Gauss-Seidel pre-conditioner as shown in eq. (3.10) is used.

$$\begin{aligned} \boldsymbol{M}_{SGS} &= (\boldsymbol{D} + \boldsymbol{E})\boldsymbol{D}^{-1}(\boldsymbol{D} + \boldsymbol{F}) \\ \Rightarrow \boldsymbol{M}_{SGS} &= \underbrace{(\boldsymbol{I} + \boldsymbol{E}\boldsymbol{D}^{-1})}_{\boldsymbol{L}} \underbrace{(\boldsymbol{D} + \boldsymbol{F})}_{\boldsymbol{U}} \\ \Rightarrow \boldsymbol{M}^{-1} &= (\boldsymbol{D} + \boldsymbol{F})^{-1}(\boldsymbol{I} + \boldsymbol{E}\boldsymbol{D}^{-1})^{-1} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1} \end{aligned} \tag{3.10}$$

Simplifying lower triangular matrix, $\boldsymbol{L}$ and upper triangular matrix, $\boldsymbol{U}$, the following equations are obtained.

$$\begin{aligned}\boldsymbol{L} = \boldsymbol{I} + \boldsymbol{E}\boldsymbol{D}^{-1} = &\begin{pmatrix} \boldsymbol{I}_{nr\times nr} & \boldsymbol{0}_{nr\times nb} \\ \boldsymbol{0}_{nb\times nr} & \boldsymbol{I}_{nb\times nb} \end{pmatrix} + \\ &\begin{pmatrix} \boldsymbol{0}_{nr\times nr} & \boldsymbol{0}_{nr\times nb} \\ \boldsymbol{E}_{nb\times nr} & \boldsymbol{0}_{nb\times nb} \end{pmatrix} \begin{pmatrix} [\boldsymbol{D}_{re}^{-1}]_{nr\times nr} & \boldsymbol{0}_{nr\times nb} \\ \boldsymbol{0}_{nb\times nr} & [\boldsymbol{D}_{bl}^{-1}]_{nb\times nb} \end{pmatrix} \\ = &\begin{pmatrix} \boldsymbol{I}_{nr\times nr} & \boldsymbol{0}_{nr\times nb} \\ [\boldsymbol{E}\boldsymbol{D}_{re}^{-1}]_{nb\times nr} & \boldsymbol{I}_{nb\times nb} \end{pmatrix} \end{aligned} \tag{3.11}$$

$$\begin{aligned}\boldsymbol{U} = \boldsymbol{D} + \boldsymbol{F} = &\begin{pmatrix} [\boldsymbol{D}_{re}]_{nr\times nr} & \boldsymbol{0}_{nr\times nb} \\ \boldsymbol{0}_{nb\times nr} & [\boldsymbol{D}_{bl}]_{nb\times nb} \end{pmatrix} + \begin{pmatrix} \boldsymbol{0}_{nr\times nr} & \boldsymbol{F}_{nr\times nb} \\ \boldsymbol{0}_{nb\times nr} & \boldsymbol{0}_{nb\times nb} \end{pmatrix} \\ = &\begin{pmatrix} [\boldsymbol{D}_{re}]_{nr\times nr} & \boldsymbol{F}_{nr\times nb} \\ \boldsymbol{0}_{nb\times nr} & [\boldsymbol{D}_{bl}]_{nb\times nb} \end{pmatrix} \end{aligned} \tag{3.12}$$

The inverse of matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ can be easily computed as

$$\boldsymbol{L}^{-1} = \begin{pmatrix} \boldsymbol{I}_{nr\times nr} & \boldsymbol{0}_{nr\times nb} \\ -[\boldsymbol{E}\boldsymbol{D}_{re}^{-1}]_{nb\times nr} & \boldsymbol{I}_{nb\times nb} \end{pmatrix} \tag{3.13}$$

$$\boldsymbol{U}^{-1} = \begin{pmatrix} [\boldsymbol{D}_{re}^{-1}]_{nr\times nr} & -[\boldsymbol{D}_{re}^{-1}\boldsymbol{F}\boldsymbol{D}_{bl}^{-1}]_{nr\times nb} \\ \boldsymbol{0}_{nb\times nr} & [\boldsymbol{D}_{bl}^{-1}]_{nb\times nb} \end{pmatrix}. \tag{3.14}$$

In the above equations, suffixes $nr \times nr$, $nr \times nb$, $nb \times nr$, $nb \times nb$ denote the dimension of the matrices where, nr, nb refer to the total number of the red and black nodes inside the domain (excluding the boundary nodes). Since the matrices are not formed explicitly, while implementing the preconditioner to any of the iterative algorithms, it is applied in two steps i.e. first by computing $\boldsymbol{w} = \boldsymbol{L}^{-1}\boldsymbol{A}$ and then by computing $\boldsymbol{z} = \boldsymbol{U}^{-1}\boldsymbol{w}$.

4. Numerical Examples

The example problems taken up for study in this section use pre - conditioner for faster convergence of the solution. The analytical functions chosen for validation are chosen in a way such that difference in the order of magnitude of the values across all the nodes is not substantial ($\sim 10^0$ to 10^2). This is to ensure that the contour plots of the numerical analysis do not result in a localized contour concentration. But, it does not restrict the applicability of the approach in any way to cases with non-zero source terms or where the difference in order of magnitudes of the values are substantial. The examples taken up in this section for validation of the proposed modified BiCGStab for elliptic PDEs assume $\beta = 1$ and $b = 0$ unless noted otherwise.

4.1. Domain with one internal boundary

The first example that is chosen is the one which has been used in the past for Immersed Interface Method and its variants[15,16,14]. In this section a detailed analysis of the algorithm is carried out. The analytical solution of eq. (3.1) for a domain with an interface is given by eq. (4.1).

$$\phi(X_1, X_3) = \begin{cases} 1 & , X_1^2 + X_3^2 < \frac{1}{4} \\ 1 + \log\left(2\sqrt{X_1^2 + X_3^2}\right) & , X_1^2 + X_3^2 \geq \frac{1}{4} \end{cases} . \tag{4.1}$$

The jumps in this case at the interface are $[\phi] = 0$ and $[\phi_n] = 2$ in a domain Ω defined by $[-1, 1] \times [-1, 1]$ (refer fig. 3).

Earlier in section 2, it is highlighted that both $\boldsymbol{r}$ and $\boldsymbol{r}^*$ need to be updated. The implication of just updating $\boldsymbol{r}$ is studied first. From fig. 4, it is evident that the residual norm converges to values as low as 10^{-1}. The solution vector thus obtained at this stage is still acceptable and was found to

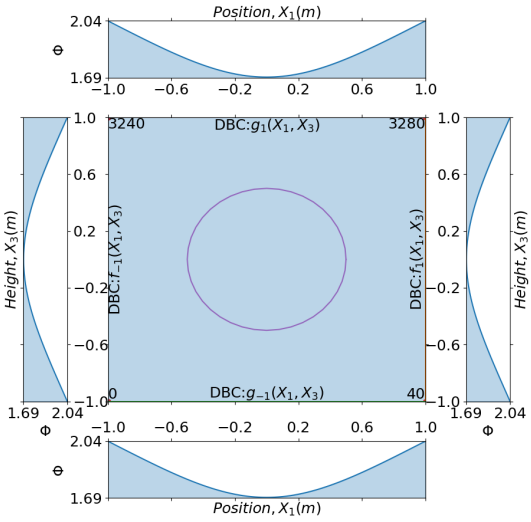


Fig. 3: Domain with boundary conditions[9]

match with analytical results[9]. But, a good algorithm should not destabilize either $\mathbf{\Phi}$ or $\boldsymbol{r}$ if further iterations are run. It is observed that the residual is not stable and if the iterations were allowed to proceed, the residual starts diverging and by the end of iteration 5000, the residual is observed to begin to diverge, reaching magnitude of 10^8. This clearly demonstrates the fact that merely correcting $\boldsymbol{r}$ is not enough and this has in fact made the BiCGStab algorithm unstable.

Now the numerical method is run again but this time $\boldsymbol{r}^*$ is updated alongside $\boldsymbol{r}$. From fig. 5, it can be seen clearly that making $\boldsymbol{r}^*$ solution dependent has stabilized the residual and it does not diverge even when the iteration number has reached as high as 10000. This signifies the importance of updating the arbitrary vector, $\boldsymbol{r}^*$. As highlighted in section 2, as the residual

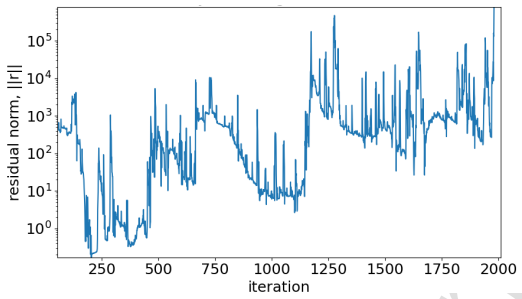


Fig. 4: Convergence of residual norm (without updating $\boldsymbol{r}^*$)

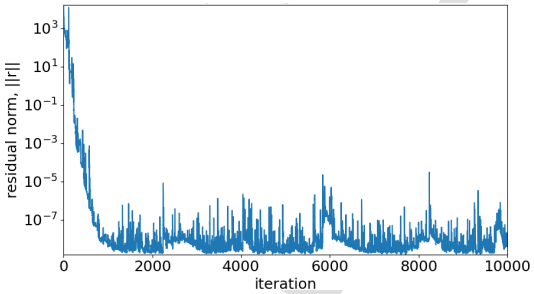


Fig. 5: Convergence of residual norm (with updating $\boldsymbol{r}^*$)

converges the correction factor $\boldsymbol{C}$ starts introducing a numerical noise. This is exhibited by the oscillations in the nature of $||\boldsymbol{r}||_2$ in fig. 5. If the correction to the residual is not applied anymore after the correction has stabilized, from fig. 6 it can be observed that the oscillations after $||\boldsymbol{r}||_2$ has approached a near-zero value are eliminated. Numerical solution of the simulation after incorporating all the modifications is presented in fig. 7.

4.1.1. Convergence and grid refinement analysis

Next a convergence and grid refinement analysis for the problem is carried out to ensure that the modified algorithm performs satisfactorily for varying

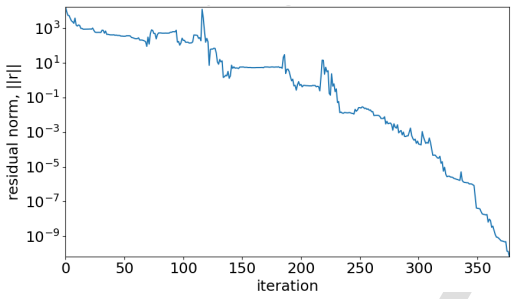


Fig. 6: Convergence of residual norm (by terminating update of $\boldsymbol{r}$ and $\boldsymbol{r}^*$ after $\boldsymbol{C}$ has stabilized)

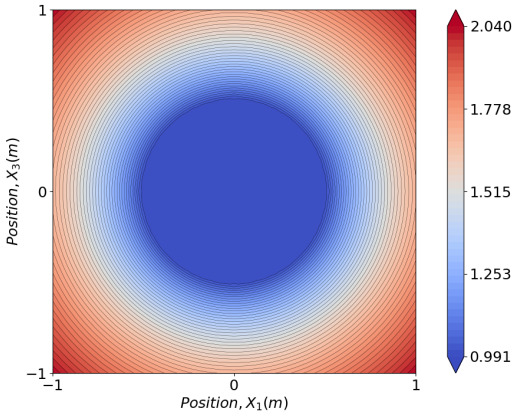


Fig. 7: Solution to the numerical analysis

Table 1: Grid refinement analysis (without updating $\boldsymbol{r}^*$)

Mesh Size ($\triangle \boldsymbol{X_i}$)	$\boldsymbol{\alpha}$	$\|\|\boldsymbol{r}\|\|$
40×40 (0.0500)	0.01	10^{-13}
80×80 (0.0250)	0.01	10^{-14}
120×120 (0.0167)	0.01	Diverges
	$0.0105 - 0.03$	Converges then Diverges
	> 0.03	Diverges
160×160 (0.0125)	0.01	Diverges
	0.011	Converges then diverges
	> 0.012	10^{-13}
200×200 (0.0100)	0.02	Converges then diverges
	0.03	Converges then diverges
	> 0.04	10^{-13}

mesh sizes. To check this, the criterion for the iterations to stop is set to $||\boldsymbol{r}||_2 \sim 10^{-9}$.

Table 1 presents the values of the order of magnitudes of the residual norm with different mesh sizes and corresponding to various values of α when only $\boldsymbol{r}$ is modified. It is evident that the value of α determines the nature of convergence of the solution as well as the number of iterations. Thus, choosing α judiciously is important to ascertain the convergence of the solution. As highlighted earlier, because of the numerical noise, choosing an appropriate value of α was difficult when $\boldsymbol{r}^*$ was not made solution dependent. However,

Table 2: Grid refinement analysis (with proposed modifications)

Mesh Size ($\triangle \boldsymbol{X_i}$)	$\boldsymbol{\alpha}$	$\|\|\boldsymbol{r}\|\|$	**Iterations**
40×40 (0.0500)	0.025	10^{-9}	1004
80×80 (0.0250)	0.025	10^{-9}	3000
120×120 (0.0250)	0.005	10^{-9}	1778
160×160 (0.0250)	0.015	10^{-9}	720
200×200 (0.0250)	0.010	10^{-9}	1265

when all the proposed modifications have been incorporated, choosing α not only became easy but the number of iterations over which the converged solution was obtained is also significantly reduced. This is clearly reflected in Table 2.

Though α is chosen depending on the situation, a thumb rule that was applied was to choose a lower value of α for a larger linear system or when the jumps are significantly higher compared to the values at boundary nodes.

4.2. Domain with multiple internal boundaries

In this section, an example is considered with multiple internal discontinuities. The usual approach to test the stability and performance of an iterative algorithm is to examine it at different levels viz.

1. Case-1: smaller value of the jumps at the interface
2. Case-2: larger value of the jumps at the interface
3. Case-3: internal boundaries governed by spatially varying functions

An analytical solution for ϕ is given by

$$\phi = (c_1 e^{pX_1} + c_2 e^{-pX_1})(c_3 \cos pX_3 + c_4 \sin pX_3) \tag{4.2}$$

where, c_1, c_2, c_3, c_4 are constants[17]. Accordingly, the following analytical function satisfying eq. (3.1) is chosen for Case-1 and Case-2.

$$\phi(X_1, X_3) = \begin{cases} (0.5e^{kX_1} + 1.5e^{-kX_1})\times & \\ (2\cos kX_3 + 7\sin kX_3) & ,\{X_1, X_3\} \in \Omega^+ \\ k_i' & ,\{X_1, X_3\} \in \Omega_i^- \end{cases} \tag{4.3}$$

where, $k = 0.009$ and Ω_i^- denotes a typical zone of discontinuity inside the global domain. Therefore, the jumps are defined as

$$\begin{aligned} [\phi]_{\Omega_i^-} &= (0.5e^{kX_1} + 1.5e^{-kX_1})(2\cos kX_3 + 7\sin kX_3) - \ k_i' \\ [\phi_n]_{\Omega_i^-, X_1} &= k(0.5e^{kX_1} - 1.5e^{-kX_1})(2\cos kX_3 + 7\sin kX_3) \\ [\phi_n]_{\Omega_i^-, X_3} &= k(0.5e^{kX_1} + 1.5e^{-kX_1})(-2\sin kX_3 + 7\cos kX_3) \end{aligned}$$

where, $[\phi]_{\Omega_i^-}$ is the jump in value of ϕ along the interface, $[\phi_n]_{\Omega_i^-, X_1}$ and $[\phi_n]_{\Omega_i^-, X_3}$ are the jumps in the derivatives in the direction normal to the vertical and the horizontal interface i.e. along X_1 and X_3 direction respectively for Ω_i^-,

A domain, Ω defined by $[150, 50]$ is considered with $\triangle X_1 = \triangle X_3 = 0.5$ (refer fig. 8). The three coloured rectangles shown inside Ω are the regions of discontinuities denoted by $\Omega_i^-; i = \{0, 1, 2\}$. The internal discontinuities, Ω_i^- are considered at arbitrary locations within the overall domain in order to understand if the spatial distribution of the discontinuities have any adverse numerical implication.

Case-1: To test this case, k_i' is considered as $\{1, 2, 3\}$ for the regions $\Omega_i^-; i = \{0, 1, 2\}$ respectively. From fig. 9 and the results presented in Table 3, it can be observed that the numerical results show an excellent match with the analytical values. The solution also converged well within an unsubstantial

Table 3: Values at a few selected coordinates

			Case-1		Case-2		Case-3	
Region	X_1	X_3	Theoretical	Numerical	Theoretical	Numerical	Theoretical	Numerical
Ω^+	10	3.5	4.2569	4.2205	4.2569	4.2205	4.2569	4.2999
	10	45.5	8.8643	8.6642	8.8643	8.6642	8.8643	8.8608
	125	3.5	4.4991	4.4783	4.4991	4.4784	4.4991	4.4851
	125	45.5	9.3686	9.1355	9.3686	9.1355	9.3686	9.1470
Ω_0^-	15.5	10.5	1	0.9673	11	10.9674	4.6228	4.9362
	18	13.5	1	0.9627	11	10.9628	4.8067	5.0148
Ω_1^-	63.5	30	2	1.6114	19	18.6111	5.6721	6.3033
	67	32.5	2	1.6708	19	18.6723	6.4565	6.7663
Ω_2^-	95	24	3	2.5814	15	14.5816	6.2780	6.0834
	97	17.5	3	2.4874	15	14.4875	6.2839	5.9546

number of iterations and which is expected given the fact that the value of the jumps are small.

Case-2: For this case, k_i' is considered as $\{11, 19, 15\}$ for the regions $\Omega_i^-; i = \{0, 1, 2\}$ respectively. From fig. 10 and the results presented in table 3, it can be observed that the numerical results are are not too far off from the analytical results implying the validity of the algorithm in such a case as well and that the modified BiCGStab is not sensitive to the values of the jumps.

Case-3: To test this case, the governing analytical function for Ω^+ is kept the same as eq. (4.3). For the domains Ω_i^-, the following analytical

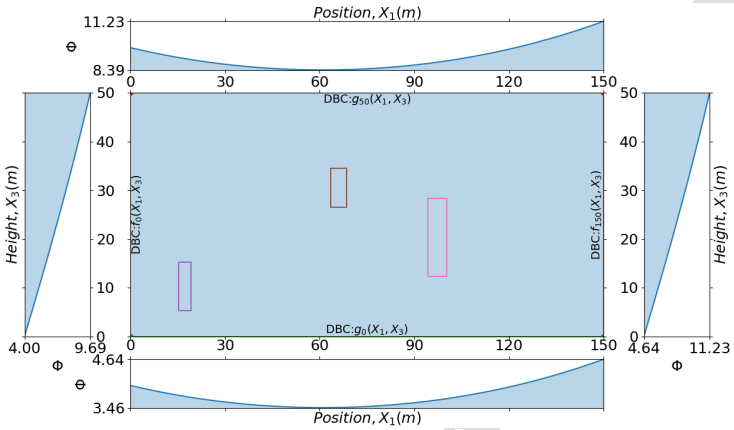


Fig. 8: Domain with multiple discontinuities

functions are assumed to satisfy eq. (3.1).

$$\phi(X_1, X_3) = \begin{cases} 1 + \log\left(2\sqrt{X_1^2 + X_3^2}\right) & , \{X_1, X_3\} \in \Omega_0^- \\ \frac{(4X_1+5)}{100} . \frac{(7X_3+9)}{100} & , \{X_1, X_3\} \in \Omega_1^- \\ 1 + \log\left(2\sqrt{X_1^2 + X_3^2}\right) & , \{X_1, X_3\} \in \Omega_2^- \end{cases} \tag{4.4}$$

The contour plot of this case is presented in fig. 11 and the results at a few selected coordinates are shown in Table 3. From the results obtained, it is evident that the algorithm is applicable for any set of arbitrary functions as long as they satisfy the governing PDE and the FDE is consistent with the governing PDE.

4.3. Single internal interface with $\beta \neq 1$ and $b \neq 0$

In all the earlier examples presented, we have considered the simplest form of elliptic PDE with $\beta = 1$ and $b = 0$. Here, we look into a more complicated situation wherein $\beta \neq 1$ and $b \neq 0$. In this example[15,18,16],

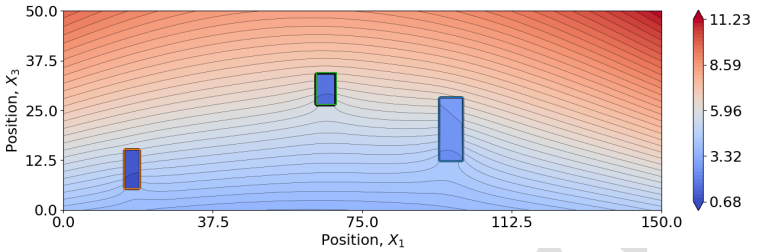


Fig. 9: Case-1: Smaller value of jumps

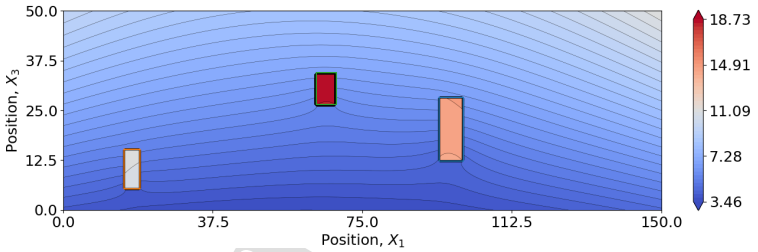


Fig. 10: Case-2: Larger value of jumps

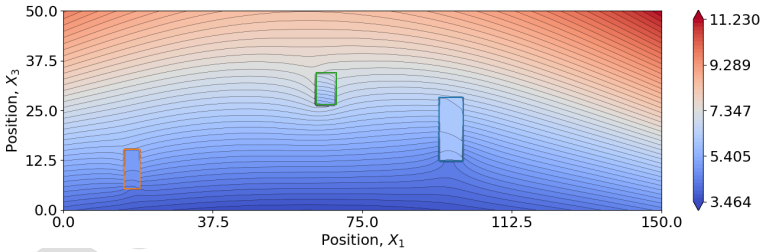


Fig. 11: Case-3: Spatially varying functions for Ω^+ and Ω^-

Table 4: Values at a few selected coordinates

Region	X_1	X_3	Theoretical	Numerical
Ω^+	-2.275	-0.925	24.3472	24.3472
	-2.65	0.375	32.9548	32.9550
	-2.2	0.525	18.3201	18.3205
	2.700	-0.575	36.7972	36.7973
Ω_0^-	-0.125	-0.475	0.2413	0.2420
	0.225	-0.425	0.2313	0.2316
	0.35	-0.35	0.2450	0.2453

$b = -(8R^2 + 4) * k_1$. The coefficient, β is given by

$$\beta(X_1, X_3) = \begin{cases} R^2 + 1 & , R < 1/2 \\ k_1 & , R \geq 1/2 \end{cases} \tag{4.5}$$

The jumps are $[\phi] = 0, [\beta\phi_n] = k_2/R$. The boundary conditions are assigned from the analytical solution,

$$\phi(X_1, X_3) = \begin{cases} R^2 & , R < 1/2 \\ (1 - \frac{1}{8b} - \frac{1}{b})/4 + (\frac{R^4}{2} + R^2) + k_2 \log(2R) & , R \geq 1/2 \end{cases} \tag{4.6}$$

where k_1, k_2 are arbitrary constants assumed as 1 and 0.1 respectively, $R = \sqrt{X_1^2 + X_3^2}$. It is observed that even for such a complicated case, the proposed modification to BiCGStab gives a numerically stable solution and the results match nicely with the analytical values which is evident from Table 4. A contour plot of the numerical results are presented in fig. 12. This example illustrates the application of the algorithm to non-homogeneous elliptic PDEs as well.

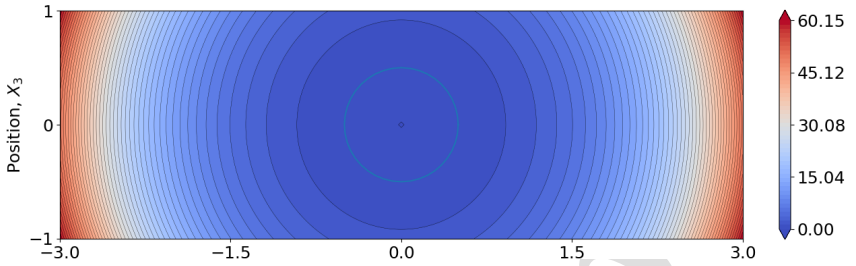


Fig. 12: Non-homogeneous elliptic PDE with discontinuous flux

5. Application to few real world problems

In this section, we look into a few real world physical problems where this algorithm can be applied.

5.1. Hele-Shaw Cell

In a Hele-Shaw cell, the fluid flows between two parallel plates separated by a very narrow-gap. Application of this flow can be found in the study of lubrication theory. Here, ϕ = pressure is the dependent variable in the homogeneous elliptic PDE and $\beta = h^3/12\mu$ (where h =spacing between the plates and μ =viscosity) is the transmissibility. It is important to note that the transmissibility is of the order of 10^{-5} in mm units which may cause a significant loss in floating point during iterations. To circumvent this issue, the FDEs are appropriately scaled by dividing with a factor of 10^{-5}. Since the governing equation is homogeneous, this scaling will not have any implication on the final numerical result. In order to validate the streamlines generated by the proposed algorithm with the experiments conducted with water[19] under hydrostatic pressure gradient, an identical Hele-Shaw cell of dimensions 610 mm by 305 mm is considered with plates separated by 0.2 mm.

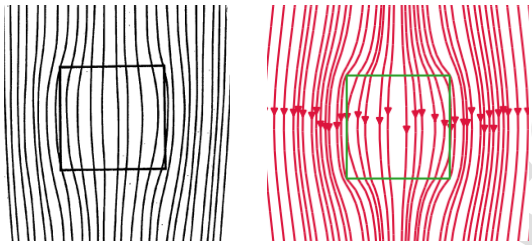


Fig. 13: Streamlines with low permeability square ($TR = 0.216$)

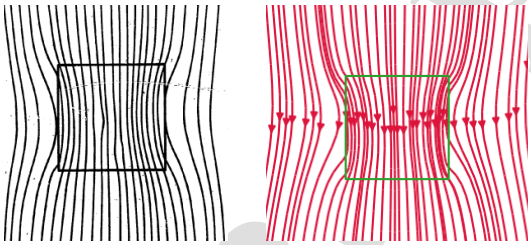


Fig. 14: Streamlines with high permeability square ($TR = 4.63$)

Four different cases are considered with different shapes of interface and different values of transmissibility ratios ($TR = \beta^-/\beta^+$). The streamlines are presented in figs. 13 to 16. The 1st figure is from experimental data and second figure is from the numerical results obtained. It can be clearly seen that the numerical results have a close match with the experimental results, thereby justifying the applicability of the proposed algorithm to a real-world scenario.

5.2. Composite materials

In this section, an application of the algorithm to composite materials is considered. It has application in determining the electrostatic and magnetostatic fields in heterogeneous media. In such kinds of problems, ϕ could be

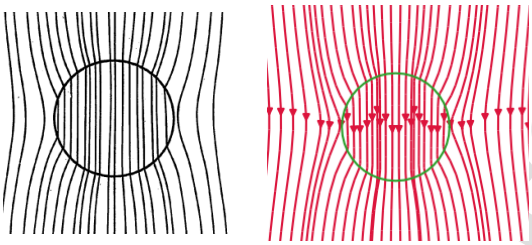


Fig. 15: Streamlines with high permeability circle ($TR = 4.63$)

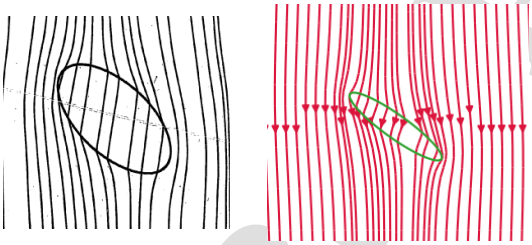


Fig. 16: Streamlines with low permeability ellipse ($TR = 0.216$)

the total potential function used to determine the electrostatic field and β could be the conductivity[20]. Researchers have considered one such problem on composite materials[14,16] where the total potential is known to follow the following function

$$\phi(X_1, X_3) = \begin{cases} \frac{2X_1}{\rho+1+s^2(\rho-1)} & , R < S \\ \frac{X_1(\rho+1)-s^2(\rho-1)X_1/R^2}{\rho+1+s^2(\rho-1)} & , R \geq S \end{cases} \tag{5.1}$$

where, $\rho = \beta^-/\beta^+$ and S is the radius of an inner circular interface. It is to be noted that ρ is mathematically identical to the transmissibility ratio defined in Hele-Shaw cell. A simple composite consisting of a periodic array of disks in a uniform background is shown in fig. 17. The reason for choosing such a problem is to study the behaviour of the algorithm when the value of ρ is substantial (5000 in the current case) and the the jumps across interface

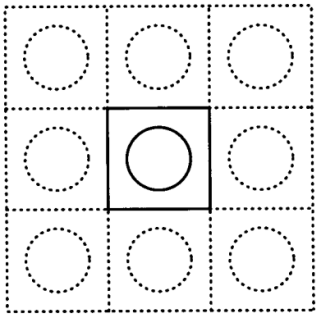


Fig. 17: A composite consisting of a periodic array of disks in a uniform background[20]

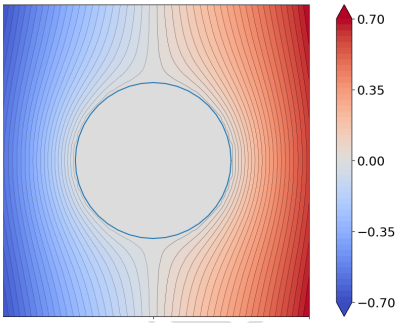


Fig. 18: Plot of total potential in the single unit highlighted in fig. 17 determined numerically

is zero (i.e. $[\phi] = 0$ and $[\beta\phi_n] = 0$), which is a special case as pointed out in algorithm 3. Even with such a high value of ρ and zero jump conditions,the results show a good match with the analytical solution, thereby justifying the applicability of the algorithm to a more realistic practical problem. Contour plot of the results obtained are presented in fig. 18.

5.3. Wind Farms

Application of this problem to a real world engineering problem i.e. in the FSI in wind farms was presented earlier by the authors[9]. However, as pointed out earlier in section 4.1, though acceptable results were obtained (both for the validation problem and the FSI problem in wind farms), the lack of correction to $||\boldsymbol{r}^*||$ or the steps taken to remove numerical noise were not implemented and as such the application presented there relied on terminating the iteration based on the ratio of the residual norm to the norm of the solution vector i.e. $||\boldsymbol{r}||_2/||\boldsymbol{\Phi}||_2 \sim 10^{-5}$. Ideally a stable algorithm

should rely solely on the residual norm such that $||\boldsymbol{r}||_2 \rightarrow 0$. In this section, the problem with one actuator disc (or one wind turbine) is presented. The jump conditions considered are defined in eq. (5.2).

$$[\phi]_f = 1/3.\overline{U}_\infty(X_3)$$
$$[\phi]_w = -1/3.\overline{U}_\infty(X_3). \tag{5.2}$$

where,

$$\overline{U}_\infty(X_3) = \frac{u_\tau}{\kappa}\left(\ln\left(\frac{X_3}{z_0}\right) + 5.75\left(\frac{X_3}{h}\right) - 1.875\left(\frac{X_3}{h}\right)^2 - \frac{4}{3}\left(\frac{X_3}{h}\right)^3 + \frac{1}{4}\left(\frac{X_3}{h}\right)^4\right) \tag{5.3}$$

κ = von Karman's constant ≈ 0.4, φ = latitude, r' = angular velocity of the earth about its own axis = $7.292 \times 10^{-5} rad/s$, C_D = drag coefficient $\approx$ 0.0014 for sea[21], u_τ = friction velocity = $M_{10}\sqrt{C_D}$; f = coriolis parameter = $2r'\sin(\varphi)$; h = gradient height = $u_\tau/6f$; H = height of the physical domain; M_{10} = wind speed at a height of 10m ; X_3 = distance in X_3 direction; z_0 = Davenport-Wieringa roughness length $\approx$ 0.0002m for sea[21]. For the simulations carried out in this paper, $M_{10} = 10m/s$ and $\varphi = 53.3498°$ (for Dublin) are assumed. After the mean velocity reaches a maximum value $\overline{U}_\infty(X_3) = (u_\tau/\kappa)(ln(h/z_0) + 67/24)$ at a height, $X_{3,max} = h$, it is assumed to become constant[22,23]. For this case, $\triangle X_1 = 1$ and $\triangle X_3 = 2$ is assumed and Ω is defined by $[2700, 974]$. The domain is shown in fig. 19 and the results of the simulation are presented in fig. 20. It is observed that the final converged solution was obtained within 2400 iterations compared to the simulation presented by the authors earlier where it took almost 4000 iterations to arrive at a minimal residual value.

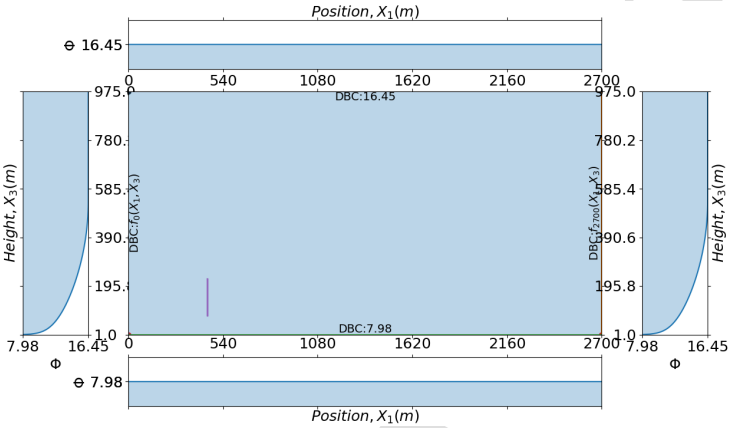


Fig. 19: A single wind turbine in a fluid domain

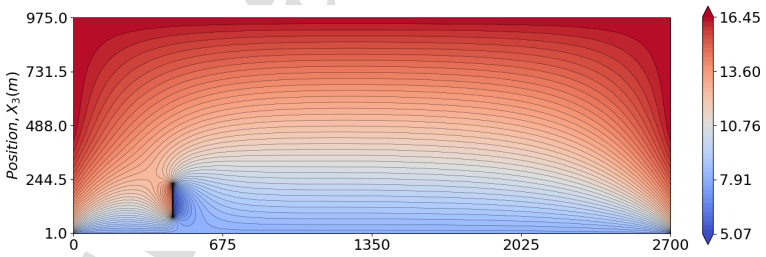


Fig. 20: Contour plot of velocity in presence of a single wind turbine

6. Performance on multiple cores

The next important aspect is to investigate the computational efficiency of the algorithm in a parallel setting. Table 5 shows the time taken per iteration when the iterative algorithm is run serially and on 2, 4 and 6 cores. It is evident that as the number of unknowns increase, there is a reduction in the processing time. When the number of unknowns are not substantial, using the maximum possible number of available cores does not essentially ascertain lower processing time. This can be seen from the case of mesh size of 80×80 where using more cores in fact increases the processing time because of an increase in the number of communication between processors relative to the number of equations in the linear system. Similarly, for mesh size of 160×160 and 200×200, 4 cores are seen to be optimal. For the real world application presented in section 5 and with the size of discretization adopted, $\mathbf{\Phi}$ is as large as 10^6. Evidently for each and every operation carried out, this operation would incur a significant computation time. It is observed that the processing time is reduced significantly with 6 cores when such a great deal of data is dealt with. Compared to a sequential process, there is almost a reduction of 60% in the time taken per iteration when 6 cores are used. One can imagine how much impact this can have in total time taken for the solution to converge. A plot showing the comparison of CPU time is given in fig. 21.

Table 5: Analysis of Computational Time

Mesh Size	**Interior Nodes**	**Time (in sec) taken per iteration for**			
		1 core	2 cores	4 cores	6 cores
40×40	1.521×10^3	$0.20 - 0.22$	$0.20 - 0.22$	$0.20 - 0.22$	$0.20 - 0.22$
80×80	6.241×10^3	$0.38 - 0.40$	$0.43 - 0.44$	$0.47 - 0.49$	$0.48 - 0.51$
120×120	1.416×10^4	$0.81 - 0.86$	$0.76 - 0.79$	$0.83 - 0.85$	$0.86 - 0.89$
160×160	2.528×10^4	$1.08 - 1.11$	$1.10 - 1.14$	$1.06 - 1.10$	$1.10 - 1.12$
200×200	3.960×10^4	$1.38 - 1.45$	$1.38 - 1.40$	$1.33 - 1.38$	$1.42 - 1.48$
2700×975	2.628×10^6	$9.75 - 10.43$	$5.82 - 6.28$	$4.15 - 4.20$	$3.67 - 3.83$

Time taken is for a Intel® Core™i9-8950HK CPU @ 2.90GHz × 12 processor.

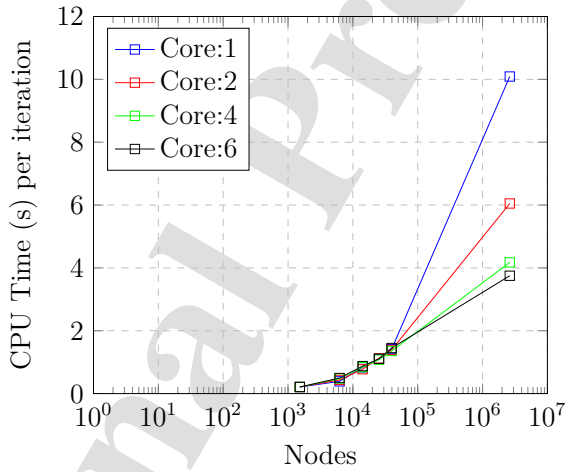


Fig. 21: Performance comparison for multi-cores

7. Conclusion

BiCGStab, as an iterative algorithm, is widely used for solutions of problems involving sparse linear systems. *However, we have shown that this*

algorithm is ineffective out-of-the-box in the case that the construction of the discrete right-hand side depends on knowledge of the true continuum solution. Realistically in many engineering problems like the FSI problem presented in this paper which requires Decomposed Immersed Interface Method, it is evident that the construction of the discrete right-hand side is continuum-solution dependent and thus BiCGStab cannot be applied straightaway. The solution to this kind of problem was originally carried out by standard red-black Gauss-Seidel iteration. However, for large linear systems with non-symmetric coefficient matrix, algorithms like BiCGStab are more efficient which motivated the current research work. Hence, in this paper, modifications to BiCGStab is proposed to arrive at a numerical solution of a system of linear equations for which the construction of the discrete right-hand side depends on the solution of the continuous problem. The validation of the modified algorithm is demonstrated by applying it to multiple fluid-structure interaction problems problems with single and multiple discontinuities. A detailed step-by-step analysis is carried out by checking the behaviour of the residual norm in order to illustrate the performance of the algorithm for every proposed modification to the algorithm. Further, a grid refinement analysis is done which clearly demonstrates that the performance of the modifications to the algorithm is not sensitive to the mesh size. Additionally the application of the algorithm to a real engineering problem has been presented to illustrate its application in the field of CFD (FSI in wind farms in section 5). The numerical analysis has been conducted for domains governed by homogeneous elliptic PDEs which have the inherent advantage of being unconditionally stable. However, this does not restrict its application

to non-homogeneous elliptic PDEs or parabolic and hyperbolic PDEs which are usually conditionally stable. Further, when one deals with large linear systems, parallel computation across multiple CPU cores become inevitable and modifications made to the iterative algorithms (BiCGStab in the current work) should not limit its applicability in such a parallel setting. This is demonstrated in section 6 where the performance of the modified BiCGStab across 1,2, 4 and 6 CPU cores is tested.

8. Acknowledgments

This work is supported by Science Foundation Ireland (SFI) grant no. 20/FFP-P/8702.

References

1. V. Voevodin, The question of non-self-adjoint extension of the conjugate gradients method is closed, USSR Comput. Math. and Math. Phys. 23 (2) (1983) 143–144. doi:10.1016/S0041-5553(83)80060-3.

2. V. Faber, T. Manteuffel, Necessary and sufficient conditions for the existence of a conjugate gradient method, SIAM J. on Numer. Analysis 21 (2) (1984) 352–362. doi:10.1137/0721026.

3. H. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. on Sci. and Stat. Comput. 13 (2) (1992) 631–644. doi:10.1137/0913035.

4. M. Hribersek, L. Škerget, Iterative methods in solving navier-stokes equations by the boundary element method, Int. J. for Numer.

Methods in Eng. 39 (1) (1996) 115–139. doi:10.1002/(SICI)1097-0207(19960115)39:1<115::AID-NME852>3.0.CO;2-D.

5. P. Gómez, J. Hernández, J. López, On the reinitialization procedure in a narrow-band locally refined level set method for interfacial flows, Int. J. for Numer. Methods in Eng. 63 (10) (2005) 1478–1512. doi:10.1002/nme.1324.

6. R. Markovinović, J. D. Jansen, Accelerating iterative solution methods using reduced-order models as solution predictors, Int. J. for Numer. Methods in Eng. 68 (5) (2006) 525–541. doi:10.1002/nme.1721.

7. Z. Mikić, E. Morse, The use of a preconditioned bi-conjugate gradient method for hybrid plasma stability analysis, J. of Comput. Phys. 61 (1) (1985) 154–185. doi:10.1016/0021-9991(85)90066-X.

8. Y. Li, R. Liu, X. Q., C. He, Z. Li, First- and second-order unconditionally stable direct discretization methods for multi-component cahn–hilliard system on surfaces, J. of Comput. and Applied Math. 401 (2022) 113778. doi:10.1016/j.cam.2021.113778.

9. S. Basu, K. Soodhalter, B. Fitzgerald, B. Basu, Flow in a large wind field with multiple actuators in the presence of constant vorticity, Phys. of Fluids 34 (10) (10 2022). doi:10.1063/5.0104902.

10. T. Gu, X. Zuo, X. Liu, P. Li, An improved parallel hybrid bi-conjugate gradient method suitable for distributed parallel computing, J. of Comput. and Appl. Math. 226 (1) (2009) 55–65. doi:10.1016/j.cam.2008.05.017.

11. Z. Ning, W. XuBen, A parallel preconditioned bi-conjugate gradient stabilized solver for the poisson problem, J. of Comput. 181 (2012) 3088–3095. doi:10.4304/jcp.7.12.3088-3095.

12. M. Riahi, Pitsbicg: Parallel in time stable bi-conjugate gradient algorithm, Applied Numer. Math. 181 (2022) 225–233. doi:10.1016/j.apnum.2022.06.004.

13. H. van der Vorst, Iterative Krylov methods for large linear systems, Vol. 13 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2009.

14. P. Berthelsen, A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions, J. of Comput. Phys. 197 (1) (2004) 364–386. doi:10.1016/j.jcp.2003.12.003.

15. R. Leveque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. on Numer. Anal. 31 (4) (1994) 1019–1044. doi:10.1137/0731054.

16. A. Wiegmann, K. Bube, The explicit-jump immersed interface method: Finite difference methods for pdes with piecewise smooth solutions, SIAM J. on Numer. Anal. 37 (3) (2000) 827–862. doi:10.1137/S0036142997328664.

17. B. Grewal, Higher engineering mathematics, Khanna Publishers, 2012.

18. Z. Li, K. Ito, Maximum principle preserving schemes for interface problems with discontinuous coefficients, SIAM J. on Scientific Comput. 23 (1) (2001) 339–361. doi:10.1137/S1064827500370160.

19. R. Greenkorn, R. Haring, H. O. Jahns, L. Shallenberger, Flow in Heterogeneous Hele-Shaw Models, Soc. of Pet. Eng. J. 4 (04) (1964) 307–316. doi:10.2118/999-PA.

20. L. Greengard, M. Moura, On the numerical evaluation of electrostatic fields in composite materials, Acta Numer. 3 (1994) 379–410. doi:10.1017/S0962492900002464.

21. R. Stull, Practical Meteorology: An Algebra-based Survey of Atmospheric Science, AVP International, University of British Columbia, 2016.

22. D. Deaves, R. Harris, A Mathematical Model of the Structure of Strong Winds, CIRIA, 1978.

23. P. Richards, S. Norris, Appropriate boundary conditions for computational wind engineering: Still an issue after 25 years, J. of Wind Eng. and Ind. Aerodyn. 190 (2019) 245 – 255. doi:10.1016/j.jweia.2019.05.012.

Declaration of Interest Statement

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The author is an Editorial Board Member/Editor-in-Chief/Associate Editor/Guest Editor for this journal and was not involved in the editorial review or the decision to publish this article.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: