A NOVEL OPTIMIZATION APPROACH TO FICTITIOUS DOMAIN METHODS

DANIEL AGRESS AND PATRICK GUIDOTTI

Abstract. A new approach to the solution of boundary value problems within the so-called fictitious domain methods philosophy is proposed which avoids well known shortcomings of other fictitious domain methods, including the need to generate extensions of the data. The salient feature of the novel method, which we refer to as SSEM (Smooth Selection Embedding Method), is that it reduces the whole boundary value problem to a linear constraint for an appropriate optimization problem formulated in a larger simpler set containing the domain on which the boundary value problem is posed and which allows for the use of straightforward discretizations. The proposed method in essence computes a (discrete) extension of the solution to the boundary value problem by selecting it as a smooth element of the complete affine family of solutions of the extended, yet unmodified, under-determined problem. The actual regularity of this extension is determined by that of the analytic solution and the choice of objective functional. Numerical experiments will demonstrate that it can be stably used to efficiently deal with non-constant coefficients, general geometries, and different boundary conditions in dimensions $d = 1, 2, 3$ and that it produces solutions of tunable (and high) accuracy.

1. Introduction

In this paper an optimization approach is proposed for the resolution of general boundary value problems within the framework of fictitious domain methods (we include so-called immersed boundary methods in this class). While the ideas and the methods readily apply to any boundary value problem, the approach will be illustrated by means of second order boundary value problems of type

$$\begin{cases}
Au = f & \text{in } \Omega, \\
Bu = g & \text{on } \Gamma = \partial \Omega,
\end{cases}$$

for an elliptic operator $A$ such as, e.g., the Laplacian $-\Delta$, and an admissible boundary operator $B$ such as, e.g., the trace $\gamma$ (Dirichlet problem), the unit outer normal derivative $\partial_{\nu}$ (Neumann problem), or a combination thereof (Robin type problem). Such boundary value problems have traditionally been strongly or weakly (when in divergence form) formulated as well-posed problems which admit a unique solution (up to a constant for some boundary conditions). Most numerical methods, reflecting this approach and viewpoint, are either a direct discretization of the problem, like in the case of finite difference methods, or the discretization of a suitable Dirichlet form based weak formulation of the problem, like in the case of finite element methods. When the domain is special, highly accurate spectral discretizations can be utilized. The former methods come with the heavy burden of generating a mesh for the domain (this becomes a serious limiting factor when dealing with some problems, like, for instance, Moving Boundary Problems or in three space dimensions), whereas the latter are limited by the small number of allowable shapes for $\Omega$ and lose some of their benefits for non-constant coefficients operators. Two widely used methods which seek to avoid these difficulties are known as the fictitious domain method and the immersed boundary method. These techniques, which we refer to simply as embedding methods, transplant the problem...
from the original domain Ω to an encompassing simple region, where straightforward discretizations
and solvers can be utilized. The approach proposed here can be viewed as a novel embedding
method, which reduces the whole boundary value problem to the role of a linear constraint to an
optimization problem for an appropriately chosen functional defined on the larger domain. The
output of the method will coincide with an approximation of the solution of the boundary value
problem in the domain Ω and with a smooth extension of it in $B \cap \Omega^c$. The degree of smoothness
will be determined by the data and the chosen functional. The method has the advantage of
working for general domains and general data (read, non-constant coefficients and any type of
boundary conditions) while delivering a paradigm to obtain, in principle, discretizations of any
degree of accuracy. Not least, it allows for a relatively straightforward robust implementation by
the use of a Preconditioned Conjugate Gradient Method (PCG). It differs from other embedding
methods in that the boundary value problem is left unmodified in the extension process to the
larger domain $B$. In other words, the interior and boundary equations are simply discretized by
means of the new regular grid in $\Omega$ and on $\partial \Omega$ for a new “extended” unknown vector defined on $B^m$
(a discretization of $B$). A solution is then computed by selecting a smooth element from the affine
space of solutions of the under-determined problem which results from imposing the equations on
the extended vector.

1.1. Description of the method. As the focus of this paper is on a numerical procedure, the
method will be described at the discrete level. A parallel continuous formulation as well as an
analysis of the method will be addressed elsewhere. The continuous counterpart, however, does
provide insights that will be exploited later in the paper in the construction of effective precondi-
tioners for the iterative solution of the derived equations. For this reason some basic properties of
the continuous operators will be mentioned here and there.

First fix a simple (square or rectangular) domain $B$ for which $\overline{\Omega} \subset B$. In this paper $B$ will
chosen to be the periodic box $(-\pi, \pi)^d \subset \mathbb{R}^d$. Denote by $B^m$ a regular uniform discretization of
$B$. Replace the continuous differential operator by a discrete counterpart $A = A^m$, defined as a
discrete evaluation of $A$ at grid-points

$$x \in \Omega^m = \Omega \cap B^m = \{x_k | k = 1, \ldots, N_m\}, N_m \in \mathbb{N},$$

where $A^m$ acts on “discrete functions” defined on $B^m$. Given a set of points

$$\Gamma^n = \{y_j | j = 1, \ldots, N_n\} \subset \Gamma$$

it is possible to discretize the boundary condition using any kind of interpolation and any kind of
discrete differentiation (where needed) based on the grid $B^m$ and obtain the corresponding discrete
equation $Bu = B^nu^n = g^n$ for the unknown vector $u^m : B^m \to \mathbb{R}$ and a discretization $g^n$ of the
boundary function $g$, defined on $\Gamma^n$. In this way the continuous boundary value problem (1.1)
can be replaced by the discrete under-determined system given by

$$Cu = C_{m,n}u^m = \begin{bmatrix} A & A^m \\ B & B^m \end{bmatrix} u = \begin{bmatrix} A^m \\ B^m \end{bmatrix} u^m = \begin{bmatrix} f^m \\ g^n \end{bmatrix} = b^{m,n} = b$$ (1.2)

where $f^m$ is a discretization of $f$ at grid points in $B^m \cap \Omega$. As the notation indicates, we shall
often suppress the superscripts and the indeces to simplify the notation. Notice that

$$u^m \in \mathbb{R}^{N_m}, f^m \in \mathbb{R}^{N_m}, \quad g^n \in \mathbb{R}^{N_n},$$

for $N_m = |\Omega \cap B^m| = |\Omega^m|$. Clearly it is always ensured that $N_m + N_n < N_n$ so that the problem,
while under-determined, admits solutions. While not strictly necessary, care is also taken to make
sure that all equations in the system are independent of each other. The reason is numerical
conditioning of the relevant matrices (more later). Now, and in contrast to available fictitious
domain methods, we don’t try to extend or modify the problem to or in the encompassing domain
Figure 1. A 1D visualization of the oscillations caused by trivial extension with no regularization. The plot only shows a region that is only slightly larger than $\Omega$ since the oscillations occur in a neighborhood of $\partial \Omega$.

Instead of using $\mathbb{B}/\mathbb{B}^m$, but rather try and find “the best” among the solutions of the under-determined problem (1.2). After all, if you use high order $\mathbb{B}^m$-based discretizations of derivatives and evaluations, the equations should be sufficient to determine a solution that achieves their order of accuracy (up to the order allowed by the regularity of the solution itself, of course).

A simple minded approach (which is fine when no regularity at all is expected) would now be to find a minimal norm solution of the problem, i.e. solve the linearly constrained optimization problem

$$\arg\min_{\{C u = b\}} \frac{1}{2} \|u\|_2^2,$$

(1.3)

where $\| \cdot \|_2$ denotes the Euclidean norm on $\mathbb{R}^{N_m}$. This would lead to the so-called normal equations and to the solution

$$u = C^T (CC^T)^{-1} b.$$

Given that the matrix $C = C_{m,n}$ consists of differential operators including the evaluation (restriction) in the domain $\Omega^m$ and on the boundary $\Gamma^n$, its transpose then corresponds to differential operators containing trivial extensions (read extensions by 0) and this leads to oscillations generated by the lack of regularity. This is made apparent in Figure 1. The “good” solution is, however, among those of the under-determined problem, and can be obtained by requiring additional regularity. As already pointed out, the discretizations $A^m$ and $B^n$ are, after all, chosen to have a desired accuracy and the truncations/trivial extensions destroy it. Thus enforcing an appropriate degree of regularity should allow for the recovery of the intrinsic accuracy of the chosen discretizations, again, compatibly with the expected regularity of the solution itself. This is also the reason for our choice to call the proposed method Smooth Selection Embedding Method (SSEM). While this selection is done in a way that is natural from the point of view of optimization [2 Chapter 10], it has a nice analytic interpretation which will greatly help with the practical implementation of the method. Let $\| \cdot \|_R$ be the discretization of a high order norm such as, for instance, $\|(1 - \Delta_p) \ell^2 / 2\|_2$, where $-\Delta_p$ denotes the periodic Laplacian on $[-\pi, \pi]^d$ and $p \geq 1$. Now the problem becomes

$$\arg\min_{\{C u = b\}} \frac{1}{2} \|u\|_R^2,$$

(1.4)
where the indeces have again been dropped for ease of reading. The constrained optimization problem (1.4) can be reformulated as the unconstrained minimization

$$\arg\min_{u \in \mathbb{R}^{N_m}, \Lambda \in \mathbb{R}^{N_\Lambda}} \frac{1}{2} \|u\|_R^2 + \Lambda^\top (Cu - b),$$

upon introduction of Lagrange multipliers $\Lambda \in \mathbb{R}^{N_\Lambda}$, where $N_\Lambda = N^{\Omega}_m + N_n$. A direct computation yields the solution

$$u = R^{-1} C^\top (CR^{-1}C^\top)^{-1} b,$$  \hspace{1cm} (1.5)

where $R$ is the (invertible) operator corresponding to the norm $\| \cdot \|_R$. Now, recalling that $C$ and $C^\top$ are truncated differential operators (more precisely containing differentiations, evaluations on subdomains, and extensions), we see that the effect of the norm is to replace the operator $C^\top$, which, upon being hit by $C$, is the cause of the oscillations in the simple minded method, by the regularized $R^{-1}C^\top$, which can be captured numerically to a higher degree of accuracy (no oscillations) when hit by $C$. While a more detailed discussion is offered later, notice that a good choice for the regularizer is a differential operator of order 4 (see Remark 1.3 below), such as $(1 - \Delta_x)^2$, since it makes the operator $CR^{-1}C^\top$ that needs to be inverted in (1.5) of order 0. This is actually only true when considering the operator $A$ in $C = [AB]^\top$ but still illustrates the idea behind the choice of appropriate regularizers and preconditioners (see Sections 2.3 and 2.4).

**Remark 1.1.** While, in the proposed method, Lagrange multipliers are introduced as they are in many a fictitious domain implementations, the approach is quite distinct from other methods. (See below, Section 1.2.) First and foremost the Lagrange multipliers are introduced for the whole problem and not only for the purpose of satisfying the boundary condition. Secondly they are introduced naturally as an enforcement tool of a linear constraint and do not require modification of the problem, the use of extensions, or the introduction of artifical terms (often in the form of sources).

**Remark 1.2.** Notice that formula (1.5) can be used as a starting point without any knowledge of a norm generating the operator $R$. One can choose any convenient regularizing operator acting on (generalized) functions defined on the box $\mathcal{B}$ instead of $R^{-1}$.

**Remark 1.3.** A “well-discretized” differential operator (with minimally smooth coefficients), meaning one with as linearly independent rows as possible, will essentially have a condition number determined by its order. This is best seen in a spectral discretization of the periodic Laplacian, which is diagonal and for which the ratio $\lambda_{\text{max}}/\lambda_{\text{min}}$ is readily available. Thus good approximations of 0-order operators will have a condition number that does not grow as the grid gets finer. This will actually be achieved for some of the implementations described below.

1.2. **Comparison with Other Embedding Methods.** Particularly relevant for this paper are the so-called fictitious domain methods and, to a lesser degree immersed boundary methods and boundary integral methods. These alternative approaches have experienced a surge in interest in recent years and seem to be particularly popular in the applied and very applied communities. Just as with the method advocated here, the fictitious domain and immersed boundary methods avoid the mesh generation step by resorting to a “container” domain of simple geometry which admits a straightforward discretization, while boundary integral methods exploit analytical knowledge about the problem to obtain a dimensional reduction by collapsing the problem to the boundary. At the heart of any of these implementations is the need to resolve the mismatch between the boundary and the simple regular grid. There is a vast literature about these methods as they can be implemented in various discretization contexts, admit a variety of distinct practical implementations within each discretization framework, and can be applied to many different boundary value problems of mathematical physics [11]. We refer to the beginning of [10] for a brief outline of many of these methods and to [5] for a concise description/numerical implementation of a number of variants.
Given the volume of publications, the choice of references made here was merely motivated by the fact that they contain a description of the methods’ philosophy and/or many useful additional references in their introduction.

1.2.1. **Fictitious Domain Methods.** A prominent implementation procedure, developed by Glowinski and coauthors in [4, 8, 7, 6] and known as the distributed Lagrange multiplier method, can be described in some more detail as follows: think of the domain $\Omega$ as a subset of a larger regular simple domain $B$, introduce a (uniform) discretization of $B$, and solve the boundary value problem by modifying the data (the right-hand-side and/or the operator $A$ in the prototypical situation considered here), usually by extending them and by introducing artificially a weighted sum of carefully chosen source terms supported outside the domain $\Omega$, i.e. in $B \setminus \Omega$, or on its boundary $\Gamma$, by determining the weights (Lagrange multipliers) so as to make sure that the boundary condition is satisfied (or at least well-approximated). We remark that a common characteristic of these techniques (and of immersed boundary methods as well) is that Neumann or Robin boundary conditions are “natural” and straightforward to include in the formulation, whereas Dirichlet boundary conditions are more challenging (see, e.g. [5]). These methods clearly have the advantage of not requiring special care nor effort in the choice of discretization for $B$. An often cited criticism of this approach is the need to extend the original elliptic operator $A$ and/or right-hand-side $f$ to corresponding objects defined on the whole of $B$. This is not always straightforward and simple minded extensions (like the trivial one by zero outside $\Omega$) introduce singularities into the problem reducing the overall accuracy of the method. See [1] regarding methods of creating smooth extensions from $\Omega$ to $B$ for the purpose of implementing fictitious domain methods. Another approach, in the context of finite elements, consists in modifying the problem’s Dirichlet form to ensure that (non-natural) boundary conditions be satisfied by possibly adding direct or more subtle penalty or penalty-like terms to it, like, e.g., the so-called Nitsche method (see [3], for example). The approach proposed here can be viewed as a novel fictitious domain method which does not require any explicit extension of the data (it can itself be used as remarked later in Section 3 to produce smooth extensions) or modification of the original boundary value problem. Moreover, it makes apparent that the real problem that any fictitious domain methods has to solve is the selection problem among the infinitely many solutions of the original problem, which are generated as the problem is viewed in a larger domain where it becomes under-determined. The direct way in which this is done here (introduction of a high order regularizer) clearly shows how the order of accuracy chosen for the interior and boundary operators can be recovered in the extended problem through an affine shift obtained by a natural (both from the point of view of PDEs and of optimization) regularization.

1.2.2. **Immersed Boundary Methods.** A very popular method used to deal with complex geometries, which is one of the motivations of this paper as well, is the so-called immersed boundary method by which a problem is extended to a simple encompassing domain admitting robust and effective discretizations. The extension is obtained by the use of Dirac distributions in the distance from the boundary (more precisely, line and surface integral distributions along the boundary) and hence typically introduces singularities which reduce the overall accuracy of the method to first order. Recently, approaches have been proposed in which the accuracy is improved by the use of extension operators that preserve smoothness. We refer in particular to [12] for an immersed boundary method which includes a smooth extension method, thereby preserving higher order accuracy, albeit at the cost of significant additional computational time (in what is called the preparation phase in the paper). We again point out that the method proposed here does not require any explicit extension since it identifies the solution among the infinitely many of the extended, under-determined problem by simply requiring smoothness in the full computational...
domain (and hence across the boundary) along with directly enforcing the PDE in \( \Omega \) and the boundary conditions on \( \partial \Omega \) by resorting only to the regular grid.

1.2.3. Boundary Integral Methods. While not directly connected to boundary integral methods, the procedure developed here allows for a nice discrete interpretation of these from the point of view of optimization. They can be used when the existence of an explicit representation for a fundamental solution \( G \) of the differential operator \( A \) is known. In this case one can use the representation 
\[
\begin{align*}
  u_h(x) &= \int_{\Gamma} G(x, y) h(y) \, d\sigma(y) \\
  &\text{for solutions of } A u = 0 \text{ and reduce the boundary value problem to determining the density } h: \Gamma \rightarrow \mathbb{R} \text{ such that }
  Bu_h(x) = B \int_{\Gamma} G(x, y) h(y) \, d\sigma(y) = g(x), \ x \in \Gamma.
\end{align*}
\]
This effectively leads to a dimensional reduction in the problem as the unknown density function is only defined on the boundary.

In formulation (1.4), this corresponds to situations where the kernel of \( A \) is known and can therefore be represented as the range of a matrix \( M \). In this case, if \( u_f \) is a particular solution of \( Au = f \), then the optimization problem can be reduced to
\[
\begin{align*}
  \min_{BMz = g - Bu_f} \frac{1}{2} \|u_f + Mz\|_R^2,
\end{align*}
\]
for the unknown (boundary and hence smaller) vector \( z \). While the regularization used here introduces an additional layer not present in a pure boundary integral formulation, the corresponding problem can also be efficiently solved given the explicit nature of the regularizer and of the encompassing domain. Clearly \( M \) corresponds to the integral operator appearing in (1.6), while \( B \) is the continuous boundary operator in (1.6) and a corresponding discretization of it in (1.7).

Remarks 1.4. We conclude this introduction with a few important remarks.

(a) The method is generic in the sense that it does require specific discretizations of the encompassing domain \( B \) and of the data. It is rather a procedure that can be adapted to the context of finite differences, finite elements, or spectral methods quite easily.

(b) It has the structure of a classical optimization problem with linear constraints for which a host of methods exists which can be used for its resolution. While there seem to be “natural” choices for the regularizing norm \( \| \cdot \|_R \), it is possible to work with other (non-quadratic) functionals, that may deliver better results for specific problems.

(c) It fully avoids the issues related to the need of generating extensions of the data from the domain \( \Omega \) to the encompassing box \( B \), while, as a matter of fact, it can itself be adapted to produce smooth extensions. See Section 3 later in this paper.

(d) As the numerical experiments presented in Section 4 will demonstrate, it is general enough to be robustly implemented for general domains, for non-constant coefficients, as well as for a variety of problems (in divergence form and not) and boundary conditions. In its high order implementations, it clearly heavily relies on the smoothness of the data (and hence of the solution), but can be used for non smooth problems as well (see Section 4.4). Clearly even better results can be obtained in this case, if specific attention is paid to the region in which the solution is singular by, e.g., introducing a weighted regularizing norm.

2. Method

2.1. Discretization of the Domain. As described in the introduction, we begin by embedding the domain \( \Omega \) into a torus \( B \) in order to make use of spectral methods and of the Fourier transform. The periodicity box \( B \) is discretized with a uniform grid \( B^m \). The boundary \( \Gamma \) is approximated with a discretization \( \Gamma^n \), which is just a set of \( n \) points lying on \( \partial \Omega \). In practice, it is best for these points to be uniformly distributed across the boundary. In two dimensions, this can be
accomplished easily by equally spacing points along an arc length parametrization of the curve. In three dimensions, equally distributing the points around a surface is more challenging, although well known algorithms exists for placing points on $S^2$. In Section 4.5, we use the well known Fibonacci algorithm (see [9]) to create a discretization.

A choice also needs to be made concerning the density of boundary points, that is, the value of $n$. When using an insufficient number of points on the boundary, the accuracy suffers, while too many points can drive up the condition number. In practice, it seems that the best results occur when three to four regular grid points lie between any two boundary points. This allows the regular grid $\mathbb{B}^m$ to easily “distinguish” the different boundary points, thereby keeping the condition number relatively low. For example, with a rectangular grid of size $m^2$, we have found that $\frac{1}{4} \frac{\pi}{\sqrt{2}}$ boundary points per unit length seems to deliver the best results. In Figure 2, we show the discretization of a disc and a flower shaped domain using such a density. For better visualization, we have only plotted the region $[-1.3, 1.3]^2$, as opposed to the entire region $[-\pi, \pi]^2$. In three dimensional problems, we have found that with a grid of size $m^3$ points, a boundary spacing of $\frac{1}{16} \frac{m}{\pi} (m^2 \pi)^2$ per unit area provides a good tradeoff between condition number and accuracy. This maintains three to four box discretization points between each boundary point along each dimension, allowing the regular box grid to resolve the “irregular” boundary discretization grid. In Figure 11, we show a discretization of the sphere with the given spacing.

2.2. Discretizing the Differential Operator. We now discuss the discretization of the differential operators $C$ and $C^T$. We recall that

$$C = \begin{pmatrix} A^m \\ B^n \end{pmatrix},$$

where $A^m$ is a matrix of evaluations of a second order differential operator at the points found in the set $\Omega^m = \mathbb{B}^m \cap \Omega$, and $B^n$ is a matrix of evaluations of a boundary operator on the finite subset $\Gamma^m$ of $\partial \Omega$. We begin by evaluating any necessary derivatives on the entire domain $\mathbb{B}^m$. As discussed in the introduction, the purpose of using a fictitious domain method is that it allows us to easily use techniques which apply to the torus, and extend them to problems with more complex geometries. In particular, the partial derivatives can be calculated using either finite difference methods or spectral methods on the torus. Spectral methods have the advantage of delivering greater accuracy for smooth problems, while finite difference methods have the advantage of being
slightly faster and being more readily applicable to a wider range of differential operators. Once
the partial derivatives have been calculated, we restrict the results to \( \Omega_m \) and multiply by the
coefficients of the operator \( A^m \).

In all of the numerical experiments below, with the exception of those in Section 4.1, we evaluate
the Laplacian used for the operator \( A^m \) spectrally. More specifically, we compute

\[
(-\Delta)^m = (\mathcal{F}^m)^{-1} \text{diag}(\{|k|^2\}_{k \in \mathbb{Z}_d^m}) \mathcal{F}^m,
\]

where \( \mathcal{F}^m \) is the discrete fast Fourier transform and \( k \in \mathbb{Z}_d^m \) is the frequency vector at discretization
level \( m \). In Subsection 4.1, where we examine nonconstant coefficients, we use the five point stencil
finite difference method. However, we would like to reiterate that \( A^m \) can implemented with any
numerical scheme for calculating derivatives on the torus; the choices we made were simply dictated
by convenience.

When applying \( (A^m)^T \), we begin by taking the transpose of the restriction operator part of \( A^m \),
which amounts to extending by 0 outside of \( \Omega_m \). We are then able to use the chosen method to
evaluate the derivatives.

Because the boundary points \( \Gamma^n \) do not lie on the regular grid, we need to use interpolation
operators when implementing the boundary operator \( B^m \). Given that we are interpolating from a
regular rectangular grid, the interpolation operators are simple. Linear, cubic, or spectral interpo-
lation can all be used. In the examples below, we have used cubic interpolation. The coefficients
of the interpolation operator should be stored as a sparse matrix to conserve memory and speed
up the computation.

**Remark 2.1.** To avoid wasting computational resources, the order of accuracy of the various
discretizations (the regularizer, the interpolation operators, and the original differential operators)
should be made to match, given that the accuracy of the solution is constrained by the accuracy
of all of these operators. In our examples, we have used cubic interpolation, whose fourth order rate
of convergence is sufficient for the fourth order rate of convergence of the eighth order regularizer.
If the expected regularity of the solution is known, it can also be taken into consideration when
making this choice.

### 2.3. The Regularizer \( R \)

As described in the introduction, the boundary value problem can be reduced to finding

\[
\arg\min_{u \in \mathbb{R}^{N_m}, \Lambda \in \mathbb{R}^{N\Lambda}} \frac{1}{2} \|u\|_R^2 + \Lambda^\top (Cu - b).
\]

Here, \( \| \cdot \|_R^2 = ||(1 - \Delta_\pi)p/2 \cdot \| \|^2 \) is a penalty seminorm introduced to enforce the regularity of the
solution across the boundary. Thus, the regularizer \( R \) imposes the \( H^p_\pi(\overline{B}) \) regularity of the solution
(if at all available, but \( p \) can be adapted to the solution). The subscript \( \pi \) indicates periodicity.
This form of the problem can then be reduced to computing

\[
u = R^{-1} C^\top \left( CR^{-1} C^\top \right)^{-1} b,
\]

where the operator \( R \) is given by

\[Ru = (1 - \Delta_\pi)^p u.\]

Because the operator \( R \) is defined over the torus \( \overline{B} \), we can use any fast solver on the rectangle to
implement \( R^{-1} \). In this paper, we will use the fast Fourier transform to take \( R^{-1} \) spectrally. We
define the matrix

\[\mathcal{R}_k = (1 + |k|^2)^{-p},\]

where \( k \in \mathbb{Z}_d^m \) is the vector of frequencies. We then note that

\[R^{-1} b = \left( \mathcal{F}^m \right)^{-1} \mathcal{R} \mathcal{F}^m b.\]
Using the fast Fourier transform, this operator can be evaluated efficiently with minimal memory requirements.

We now turn to discussing which power $p$ is optimal to use for the regularizer. As described in the introduction, a regularizer of order $p$ seeks to find an $H^p$ extension of the solution. Thus, whenever the true solution is smooth, we expect that the rate of convergence of the discrete solution will be of order $p$. In Figure 3, we demonstrate the rate of convergence of various order regularizers. The problem studied is posed on the disc $D$ of radius 1 and reads

$$\begin{align*}
-\Delta u &= -3(x - y) \quad \text{in } D, \\
u &= x^3 - y^3 \quad \text{on } \partial D.
\end{align*}$$

The exact solution is $u = x^3 - y^3$. Figure 3 shows the $L_2$ and $L_\infty$ convergence rates plotted against $2^m$, the size of the grid. The $p$ rate of convergence is very apparent for each regularizer.

![Figure 3. Convergence of different order regularizers](image)

We conclude that it is advantageous to use as high an order as the regularity of the solution suggests for the regularizer (whenever possible, since there are some practical limitations as we shall see later). However, as demonstrated below, increasing the order of the regularizer increases the order of the operator, and therefore increases its condition number. If an operator is of order $p$, the condition number will grow like a polynomial of degree $p$ as the grid size increases. For example, on a grid of size $m$, the largest eigenvalue of the Laplace operator will be of size $m^2$ (see Remark 1.3). With a good choice of preconditioner, these large condition numbers can be significantly improved. However, preconditioning is more effective for lower order regularizers. Thus, a balance must be struck between accuracy and condition number. In practice, we have found that, for a second order problem, the fourth and sixth order regularizers can be used with grids of any size, whereas higher order smoothers can be only be used for coarser grids, given the preconditioners we have been able to find. Notice that this issue is mute if we choose to use direct inversion of the matrices, which, however, can become impractical.

The order four regularizer has the particular advantage that, with the preconditioning discussed in Section 2.4, the condition number of the corresponding operator remains uniformly bounded, regardless of grid size. This is because the operator $C$ is second order in the interior. Thus, when $R^{-1}$ is of order $-4$, $CR^{-1}C^T$ is of order 0 in the interior. In Table 1, we show the growth in...
condition number for the 4th, 6th, and 8th order regularizers ($p = 2, 3, 4$, respectively) for the discretization of the disc problem described in Section 2.1.

| Grid Points | Boundary Points | Condition Number | CPU Time |
|-------------|-----------------|------------------|----------|
|             |                 | $p = 2$ | $p = 3$ | $p = 4$ | $p = 2$ | $p = 3$ | $p = 4$ |
| $16 \times 16$ | 4 | 3 | 16 | 56 | 0.01 | 0.01 | 0.01 |
| $32 \times 32$ | 8 | 6 | 20 | 79 | 0.01 | 0.02 | 0.03 |
| $64 \times 64$ | 16 | 7 | 22 | 149 | 0.02 | 0.05 | 0.11 |
| $128 \times 128$ | 32 | 6 | 25 | 301 | 0.11 | 0.21 | 0.47 |
| $256 \times 256$ | 64 | 6 | 25 | 594 | 0.77 | 1.8 | 5.93 |

Table 1. Condition numbers/CPU times of different regularizers. CPU times are on an Intel 7700HQ.

2.4. Preconditioning and PCG Implementation. In general, the condition number of the operator $CR^{-1}C^T$ is too large to allow us to use iterative linear system solution methods. This occurs because the boundary operator and the interior operator have different orders. To demonstrate this, we think of the operator $CR^{-1}C^T$ as a block matrix

$$CR^{-1}C^T = \begin{pmatrix} A^n R^{-1}(A^m)^T & A^n R^{-1}(B^n)^T \\ B^n R^{-1}(A^m)^T & B^n R^{-1}(B^n)^T \end{pmatrix} = \begin{pmatrix} C_1 & C_2 \\ C_2^T & C_3 \end{pmatrix}.$$

If $R^{-1}$ is an operator of order $-k$, then the matrix $C_1$ is of order $4 - k$, $C_2$ is of order $2 - k$ and $C_3$ is of order $-k$ (for a boundary operator of order 0). The large order together with the mismatch in scaling causes a very large condition number. We will describe a simple preconditioner which works effectively for the fourth, sixth, and eighth order regularizers. The preconditioning consists of finding approximate inverses to the $C_1$ and $C_3$ blocks independently. The general philosophy is to precondition the operator so that it becomes order 0.

We begin by finding an approximate inverse for the $C_3$ block. In the following description, we will consider a Dirichlet problem, where the boundary operator $B$ consists of evaluation on the boundary. However, an analogous procedure can be used to construct a preconditioner for any boundary condition. In the following discussion, the discrete boundary points belonging to $\Gamma^n$ will be denoted by $y_i$ for $1 \leq i \leq n$. We recall that

$$B^n : \mathbb{R}^B \to \mathbb{R}^{\Gamma^n} \text{ and } R : \mathbb{R}^B \to \mathbb{R}^B.$$

We now consider the operators

$$\tilde{B}^n : C(\mathbb{B}) \to \mathbb{R}^{\Gamma^n} \text{ where } [\tilde{B}^n u]_i = \langle \delta_{y_i}, u \rangle = u(y_i)$$

and

$$\tilde{R} : H_2^{2p-d/2-\epsilon}(\mathbb{B}) \to H_2^{d/2-\epsilon}(\mathbb{B}) \text{ where } \tilde{R} u = (1 - \Delta_\pi)^p u.$$

We note that $\tilde{B}^n$ and $\tilde{R}$ can be viewed as approximations of $B^n$ and $R$ respectively, operating on the continuous $\mathbb{B}$ rather than the discrete $\mathbb{B}$. The integral operator

$$\tilde{C}_3 := \tilde{B}^n \tilde{R}^{-1}(\tilde{B}^n)^T : \mathbb{R}^{\Gamma^n} \to \mathbb{R}^{\Gamma^n} \text{ with kernel } [\tilde{C}_3]_{ij} = \langle \delta_{y_i}, \tilde{R}^{-1} \delta_{y_j} \rangle, \ 1 \leq i, j \leq n$$
is then a good approximation of $C_3$. Notice that $\delta_y \in H^{-d/2-\varepsilon}(\mathbb{B})$ for any $y \in \mathbb{B}$ and $\varepsilon > 0$. If we define

$$h(y) = \left( \tilde{R}^{-1} \delta \right)(y)$$

as the fundamental solution of $\tilde{R}$ on the torus $\mathbb{B}$, we find, by translation invariance of the torus, that

$$[\tilde{C}_3]_{ij} = \left( \frac{2\pi}{m} \right)^d h(y_i - y_j).$$

Here, by an abuse of notation, the factor $\left( \frac{2\pi}{m} \right)^d$ is built-in to account for the fact that the “matrix” $\tilde{C}_3$ acts as an integral operator and not as simply matrix-vector multiplication. Given a good value table for $h$, we can easily calculate the matrix $\tilde{C}_3$ by evaluating the function $h$ on the matrix of differences between the points in $\Gamma^n$. Given the explicit matrix $\tilde{C}_3$, we can directly calculate $(\tilde{C}_3)^{-1}$ and use it as a preconditioner for the $C_3$ block of the matrix. Although this entails inverting a dense matrix, for coarse grids in three dimensions and even for very fine grids in two dimensions, the number of boundary points is small enough that inverting, storing, and applying the matrix is computationally negligible.

To calculate the function $h$, several methods can be used. In our implementation, we proceed as follows. We take $\tilde{m}$ large and generate a very fine grid of size $\tilde{m}^d$ on the torus $\mathbb{B}$. In our examples, we used $\tilde{m} = 4096$. We define the vector $\delta_{\tilde{m}}$ by

$$\delta_k^\tilde{m} = \begin{cases} \tilde{m}^d (2\pi)^d, & \text{if } k = 0, \\ 0, & \text{otherwise.} \end{cases}$$

The vector $\delta_{\tilde{m}}$ is then an approximation of the continuous (periodic) $\delta$ distribution supported in the origin. We then compute $R^{-1} \delta_{\tilde{m}}$ on the fine grid. This function is a good approximation of $h$ evaluated at the points $\mathbb{B}_{\tilde{m}}$. We use cubic interpolation to evaluate $h$ at points which do not lie in $\mathbb{B}_{\tilde{m}}$.

We now turn to finding an approximate inverse to $C_1$. The matrix $C_1$ depends on the order of the regularizer we have chosen. For the fourth order regularizer, we note that the matrix $C_1$ is of order $0$. Thus, no preconditioning is necessary, and $\tilde{C}_1^{-1}$ can be taken as the identity. For the higher order regularizers, we note that the operator $C_1$ is the discretization of a differential operator of order $4 - k$. We wish to precondition in such a way as to reduce the order of the operator to order $0$. Thus, we define the preconditioner

$$\tilde{C}_1^{-1} u = (1 - \Delta_\Omega)^{k-4} u.$$ 

Here, $\Delta_\Omega$ is the Laplace operator on $\Omega$. In order to implement it, we use the domain discretization $\Omega^m = \mathbb{B}^m \cap \Omega$ and a finite difference scheme to discretize the Laplacian on $\Omega^m$. In the examples the five points stencil (seven points in three dimensions) was chosen to take the Laplacian on $\Omega^m$. With this preconditioner

$$\tilde{C}_1^{-1} = \begin{pmatrix} C_1^{-1} & 0 \\ 0 & \tilde{C}_3^{-1} \end{pmatrix},$$

the condition number of the fourth order regularizer stays uniformly bounded, independent of grid size, the condition number of the sixth order regularizer grows slightly with grid size, and the condition number of the eighth order regularizer grows significantly with grid size; we refer to Table 1. Thus, the fourth and sixth order regularizers are effective for both coarse and fine grids in two and three dimensions, while the eighth order regularizer is more practical for coarse grids, particularly in three dimensions; see Table 6 for CPU times in three dimensions. We are continuing to investigate whether the preconditioner for the the eighth order regularizer can be improved to allow for usage on very fine grids. This preconditioning procedure allows us to use
the PCG method to obtain the solution. We remark that this can be implemented with minimal memory requirements. The differential operators and the regularizers are implemented with finite difference and spectral methods and require only minimal storage of matrix values. We also note that, given that the most computationally expensive part of our procedure is the FFT, the program is simple to parallelize using off the shelf FFT libraries. However, even without parallelization, the method is quite efficient. Table 1 shows the CPU times for the algorithms all executed in Python using the Numpy library on an Intel i7-7700HQ.

Next we come back to an observation we made in the introduction according to which the methodology developed here to solve boundary value problems, can actually be used to compute smooth extensions of function defined in $\Omega^m$ to the whole periodic box $\mathbb{B}^m$. This is done in the next subsection.

3. Extension Problems

As described in the introduction, a common problem that is encountered when embedding a problem with complex geometry in a container space is the (smooth) extension of some or all of the data from the original domain to the encompassing one. We briefly outline how the proposed method can be used to generate periodic $H^p(\mathbb{B})$ extensions to the torus. Other boundary conditions on $\mathbb{B}$ can also be used, of course, with the appropriate modifications. In this paper we stick to the periodic setting.

The extension problem consists of finding $\tilde{u} \in H^p(\mathbb{B})$, given a domain $\Omega \subset \mathbb{B}$ as well as a function $u \in H^p(\Omega)$, satisfying $\tilde{u}|_{\Omega} = u$. Following the spirit of the proposed method, we begin by taking a regular discretization $\mathbb{B}^m$ of $\mathbb{B}$. We then define the operator $C$ to be the restriction to $\Omega^m = \Omega \cap \mathbb{B}^m$. Correspondingly $C^T$ is simply given by the extension by 0 from $\Omega^m$ to $\mathbb{B}^m$. Finally we look for $\tilde{u}$ which minimizes the energy defined as

$$\arg\min_{u \in \mathbb{R}^N, \Lambda \in \mathbb{R}^N} \frac{1}{2} \|u\|_R^2 + \Lambda^T (Cu - b),$$

As above, we take $\|u\|_R^2 = \|(1 - \Delta_{\pi})^p u\|^2$. The problem reduces to

$$u = R^{-1} C^T (CR^{-1} C^T)^{-1} b.$$ (3.8)

The preconditioning of the system follows that of Section 2.4. As above, we try to bring the operator to order 0. As $C$ is already an order 0 operator and $R^{-1}$ is of order $-2p$, the correct preconditioner is $(1 - \Delta_{\Omega})^p$. Here, we evaluate $(1 - \Delta_{\Omega})^p$ by using a finite difference method on $\Omega$, as discussed in Section 2.4.

We remark that the different extensions resulting from the different choice of $p$ used in the regularizer will produce functions of a very different nature, as they are minimizing different powers of the Laplacian; which power $p$ is optimal will depend on the application which the extension is being used for.

Next let’s look at an example. Let $\Omega$ be the unit disc. We will produce an extension of the function $u|_{\Omega} = \frac{1}{4}(1 - r^2)$. We do this by setting $Ru = (1 - \Delta_{\pi})^p u$, where $p = 2, 3, 4$. The construction of $R^{-1}$ is the same as in Section 2.3. For the preconditioner, we use a five point stencil finite difference method of the appropriate order described above. The result of the extension done on a $128 \times 128$ grid using the three regularizers are shown in Figure 4 with a graph and in Figure 5 as a contour plot. In Table 2 we show how the different regularizers affect different Sobolev seminorms of the corresponding minimizers.
4. Numerical Experiments

We now present a number of numerical experiments to demonstrate the versatility of our method. In each experiment, we will describe the grid and the method of discretization of the operator $C$. We will then solve the system using the PCG method, using the methodology and the preconditioning described in Section 2. We will then record the $L_2$ and $L_\infty$ errors, and the number of iterations to convergence for the fourth, sixth, and eighth order regularizers. Although we shall only go to a grid size of $512^d$, the method is fast enough to solve on significantly larger grids, particularly for the fourth and sixth order regularizers. We will consider a problem with nonconstant coefficients, a problem with complex (star-shaped) geometry, Neumann boundary conditions, a problem with nonsmooth boundary conditions and a three dimensional problem with $\Omega = B^3$, the ball of radius 1.

4.1. Nonconstant Coefficients. We again pose the problem in unit the disc with the discretization as described in 2 and study the problem

$$
\begin{cases}
-((2 + y)\partial_x^2 + (2 - x)\partial_y^2)u &= -6x(2 + y) + 6y(2 - x) \\
 u|_{\partial\Omega} &= x^3 - y^3.
\end{cases}
$$
The actual solution is \( x^3 - y^3 \). In this case, we use the five point stencil finite difference scheme for the interior differential operator. All else is carried out just as in the example in Section 2.

Comparing Figures 3 and 6, we see that the accuracy achieved is roughly equivalent for both the constant coefficient problem and the nonconstant coefficient problem. However, as shown by Table 3, the condition number grows faster for the nonconstant coefficient problem. In particular, in the nonconstant coefficient case, the condition number does grow even for the fourth order regularizer, albeit very slowly. Further study is needed to see if preconditioners can be tailored to specific non-constant coefficient problems to get better results. However, the example does show that the method is robust enough to efficiently solve problems with nonconstant coefficients.

![L_2 Error-Different Regularizers](image1)

**Figure 6.** Convergence of different order regularizers for non-constant coefficients.

| Grid Points | Boundary Points | PCG Iterations | CPU Time |
|-------------|-----------------|----------------|----------|
|             |                 | \( p = 2 \)    | \( p = 3 \) | \( p = 4 \) |
| 16 \times 16| 4               | 14             | 14       | 14       |
| 32 \times 32| 8               | 34             | 41       | 54       |
| 64 \times 64| 16              | 48             | 61       | 102      |
| 128 \times 128| 32            | 55             | 74       | 155      |
| 256 \times 256| 64            | 61             | 87       | 237      |
| 512 \times 512| 128           | 65             | 93       | 400      |

**Table 3.** PCG iterations/CPU times of different regularizers - nonconstant coefficients. CPU times are on an Intel 7700HQ.

4.2. **A Star Shaped Domain.** For a problem on a more complex domain, we consider a five petalled flower.

\[
\Omega = \{ (r, \theta) | r < 1 + .2 \cos(5\theta) \}.
\]
Figure 2 shows $\Omega$ with its boundary discretization. We solve the problem
\[
\begin{cases}
-\Delta u = 0 & \text{in } \Omega, \\
u = x^2 - y^2 & \text{on } \partial \Omega
\end{cases}
\]
The true solution is $u = x^2 - y^2$. We will use spectral methods to take the Laplacian on the interior. All else is carried out as in the previous examples. This experiment demonstrates that the accuracy of the method is maintained even for a complex geometry, as shown in Figure 7. However, Table 4 shows that the condition number grows faster for the higher order regularizers in this case. We believe that, given the complexity of the shape, boundary points are necessarily closer together and therefore more difficult for the interior grid to resolve. However, the method is still quite robust, independent of domain shape.

![L2 Error-Different Regularizers](image1)

**Figure 7.** Convergence of different order regularizers in the star shaped domain.

| Grid Points | Boundary Points | PCG Iterations $p = 2$ | PCG Iterations $p = 3$ | PCG Iterations $p = 4$ | CPU Time $p = 2$ | CPU Time $p = 3$ | CPU Time $p = 4$ |
|-------------|----------------|------------------------|------------------------|------------------------|----------------|----------------|----------------|
| 16 \times 16 | 5              | 14                     | 24                     | 25                     | 0.01           | 0.03           | 0.03           |
| 32 \times 32 | 10             | 25                     | 46                     | 88                     | 0.03           | 0.06           | 0.18           |
| 64 \times 64 | 20             | 34                     | 76                     | 206                    | 0.05           | 0.18           | 0.55           |
| 128 \times 128 | 40           | 35                     | 118                    | 430                    | 0.2            | 0.75           | 2.92           |
| 256 \times 256 | 80            | 35                     | 141                    | 744                    | 1.36           | 7.1            | 40.34          |
| 512 \times 512 | 160           | 35                     | 153                    | 1481                   | 7.52           | 41.2           | 389.71         |

**Table 4.** PCG iterations/CPU times of different regularizers - star shaped domain. CPU times are on an Intel 7700HQ.
4.3. Neumann Boundary Conditions. In order to demonstrate how the method is also applicable to other boundary conditions, a Neumann problem is considered. We will work on the disc of radius one centered in the origin, using the discretization described in Section 2.1 and shown in Figure 2. The exact problem reads

\[
\begin{align*}
\Delta u &= 0 \quad \text{in } \Omega, \\
\frac{\partial u}{\partial \nu} &= 2(x^2 - y^2) \quad \text{on } \partial \Omega.
\end{align*}
\]

The analytic solution is \( u = x^2 - y^2 \). We take the derivative in the interior using the spectral Laplacian, and we evaluate the normal derivative by taking a central difference followed by cubic interpolation. We subtract \( u(0,0) \) to eliminate the constant functions in the kernel. The convergence results are displayed in Figure 8 and the CPU times are recorded in Table 5. We note that the rate of convergence in the Neumann problem is one degree slower than in the Dirichlet problem.

![Figure 8](image-url)  
**Figure 8.** Convergence rates for different order regularizers - Neumann problem.

| Grid Points | Boundary Points | PCG Iterations | CPU Time |
|-------------|-----------------|----------------|----------|
|             |                 | \( p = 2 \) | \( p = 3 \) | \( p = 4 \) | \( p = 2 \) | \( p = 3 \) | \( p = 4 \) |
| 16 \( \times \) 16 | 4 | 4 | 4 | 4 | 0.01 | 0.01 | 0.01 |
| 32 \( \times \) 32 | 8 | 8 | 13 | 16 | 0.01 | 0.02 | 0.02 |
| 64 \( \times \) 64 | 16 | 15 | 25 | 37 | 0.02 | 0.05 | 0.08 |
| 128 \( \times \) 128 | 32 | 38 | 45 | 78 | 0.16 | 0.36 | 0.42 |
| 256 \( \times \) 256 | 64 | 85 | 97 | 158 | 3.25 | 4.85 | 6.27 |
| 512 \( \times \) 512 | 128 | 188 | 207 | 488 | 32.93 | 43.61 | 102.83 |

**Table 5.** PCG iterations/CPU times of different regularizers - Neumann problem. CPU times are on an Intel 7700HQ.
4.4. **Non-regular Problem.** Although our method is by its nature more suited to smooth problems, it is not strictly limited to them. While the method in its current form has no hope of properly approximating the solution in the immediate vicinity of a singularity, outside a small ball containing the singularity, it converges reasonably well to the solution. In the following example, we let \( \Omega \) be the unit disc and study the nonsmooth problem

\[
\begin{aligned}
-\Delta u &= 0 \quad \text{in } \Omega, \\
u &= g \quad \text{on } \partial \Omega,
\end{aligned}
\]

where

\[
g(\theta) = \begin{cases}
1 & \text{if } 0 \leq \theta \leq \pi \\
-1 & \text{if } \pi \leq \theta \leq 2\pi
\end{cases}
\]

in polar coordinates. In solving this problem, we will use the same discretization of the domain and operators used in the example studied in Section 2. We note that the higher order regularizers do not provide an advantage when the solution itself is not smooth, so we restrict ourselves to the fourth order regularizer. The true solution of this boundary value problem can be given in the form of the series

\[
u(r, \theta) = \sum_{k=1}^{\infty} g_k r^k \sin(k \theta),
\]

where the Fourier coefficients \( g_k \) are defined by

\[
g_k = \begin{cases}
\frac{4}{k\pi} & \text{if } k \text{ is odd}, \\
0 & \text{if } k \text{ is even}.
\end{cases}
\]

The two singularities occur at \( y_1 = [1, 0] \) and \( y_2 = [-1, 0] \). We will study the solution away from the singularities in two ways. First, we will look at the \( L_2 \) and \( L_\infty \) errors on \( \tilde{\Omega} = \Omega \setminus (B(y_1, 0.2) \cup B(y_2, 0.2)) \), which cuts out the singularities. The region \( \Omega \) and the corresponding errors are shown in Figure 9. The graphs roughly show first order convergence of the error. We also show the approximated solution along the curve \( r = .9, 0 \leq \theta \leq \pi \) in Figure 10. As the grid becomes more dense, the approximations get closer to the true solution.

We also would like to point out that the general framework of our method could potentially be modified to allow it to deal with singular problems more effectively; this could either be done by allowing an adaptive grid which is more dense in the region of the singularity or by modifying the norm used to generate the regularizer \( R^{-1} \) by introducing weights or allowing for some singular behavior.

![Figure 9. Convergence of the nonregular problem away from the singularities.](image-url)
4.5. The Sphere. For a three dimensional example, we choose the unit sphere embedded in the three dimensional torus. The boundary is discretized by the well known Fibonacci lattice \[9\] which comes close to distributing points uniformly on the sphere. With a box discretization of \(m^3\) points, we will use \(\frac{m^2}{16\pi}\) boundary points; see Section 2.1 for justification and Figure 11 for an image. The problem considered is

\[
\begin{cases}
-\Delta u = 1 & \text{in } \Omega, \\
u = 0 & \text{on } \partial\Omega.
\end{cases}
\]

The actual solution is \(u = \frac{1 - r^2}{6}\). We see in Figure 12 that the accuracy achieved is similar to that of the two dimensional problem. Also, the condition number stays uniformly bounded when \(p = 2\), similar to the two dimensional case. However, as evidenced in Table 6, for the higher order regularizers, the condition number grows significantly faster, given the larger number of both interior and boundary points. Thus, we believe it will be greatly beneficial to find a better preconditioner which can control the condition number of the higher order regularizers.

[Figure 10. Values of the approximated solution along the curve \(r = .9\).]

[Figure 11. Discretizing the boundary of a sphere]
Figure 12. Convergence rates of different regularizers on the sphere.

| Grid Points | Boundary Points | PCG Iterations | CPU Time |
|-------------|-----------------|----------------|----------|
|             |                 | \( p = 2 \)     | \( p = 3 \) | \( p = 4 \) | \( p = 2 \) | \( p = 3 \) | \( p = 4 \) |
| 16 \times 16 | 5               | 16             | 23        | 36        | 0.41       | 0.19       | 0.34       |
| 32 \times 32 | 20              | 20             | 34        | 69        | 0.16       | 0.29       | 0.76       |
| 64 \times 64 | 81              | 26             | 39        | 76        | 2.53       | 5.15       | 12.01      |
| 128 \times 128 | 325          | 29             | 41        | 248       | 23.92      | 39.81      | 375.17     |
| 256 \times 256 | 1303         | 29             | 60        | 2220      | 225.3      | 622.77     | 21853.86   |

Table 6. PCG iterations/CPU times of different regularizers on the sphere. CPU times are on an Intel 7700HQ.

References

[1] J. P. Boyd. Fourier embedded domain methods: extending a function defined on an irregular region to a rectangle so that the extension is spatially periodic and \( C^\infty \). Applied Mathematics and Computation, 161(2):591–597, 2005.
[2] S. Boyd and L Vandenberghe. Convex Optimization. Cambridge University Press, 2004.
[3] E. Burman and P. Hansbo. Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method. Applied Numerical Mathematics, 62(4):328–341, 2012.
[4] E. J. Dean, Q. V. Dinh, R. Glowinski, J. He, T. W. Pan, and J. Périaux. Least squares/domain imbedding methods for Neumann problems: Applications to fluid dynamics. In Fifth Internat. Symposium on Domain Decomposition Methods for Partial Differential Equations, pages 451–475, 1992.
[5] S. Del Pino and O. Pironneau. A fictitious domain based general PDE solver. Numerical Methods for Scientific Computing Variational Problems and Applications, Barcelona, 2003.
[6] R. Glowinski and Q. He. A least-squares/fictitious domain method for linear elliptic problems with Robin boundary conditions. Communications in Computational Physics, 9(3):587–606, 2011.
[7] R. Glowinski and Y. Kuznetsov. Distributed Lagrange multipliers based on fictitious domain method for second order elliptic problems. Computer Methods in Applied Mechanics and Engineering, 196(8):1498–1506, 2007.
[8] R. Glowinski, T. W. Pan, and J. Périaux. A fictitious domain method for Dirichlet problem and applications. Computer Methods in Applied Mechanics and Engineering, 111(3-4):283–303, 1994.
[9] Á. González. Measurement of areas on a sphere using Fibonacci and latitude–longitude lattices. *Mathematical Geosciences*, 42(1):42–49, 2010.

[10] X. Li, J. Lowengrub, A. Rätz, and A. Voigt. Solving PDEs in complex geometries: a diffuse domain approach. *Communications in Mathematical Sciences*, 7(1):81–107, 2009.

[11] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[12] D. B. Stein, R. D. Guy, and B. Thomases. Immersed boundary smooth extension: A high-order method for solving PDE on arbitrary smooth domains using Fourier spectral methods. *Journal of Computational Physics*, 304:252–274, 2016.

University of California, Irvine, Department of Mathematics, 340 Rowland Hall, Irvine, CA 92697-3875, USA

E-mail address: dagress@uci.edu and gpatrick@math.uci.edu