Yue Hao and Valeria Simoncini*

Matrix equation solving of PDEs in polygonal domains using conformal mappings

https://doi.org/10.1515/jnma-2020-0035
Received April 28, 2020; revised October 29, 2020; accepted October 29, 2020

Abstract: We explore algebraic strategies for numerically solving linear elliptic partial differential equations in polygonal domains. To discretize the polygon by means of structured meshes, we employ Schwarz–Christoffel conformal mappings, leading to a multiterm linear equation possibly including Hadamard products of some of the terms. This new algebraic formulation allows us to clearly distinguish between the role of the discretized operators and that of the domain meshing. Various algebraic strategies are discussed for the solution of the resulting matrix equation.

Keywords: elliptic partial differential equations, Schwarz–Christoffel mapping, matrix equations, iterative methods

Classification: 65F10, 65N22

1 Introduction

We are interested in exploring matrix-oriented algebraic strategies for numerically solving two-dimensional partial differential equations (PDEs) of the type

$$-\alpha_1 u_{xx} - \alpha_2 u_{yy} + \mathbf{w}^T \cdot \nabla u + \beta u = f, \quad (x, y) \in \mathscr{P} \tag{1.1}$$

where $u = u(x, y)$, $f = f(x, y)$, $\mathscr{P} \subset \mathbb{R}^2$ is the interior of a polygon, the vector $\mathbf{w}^T = (w_1(x, y), w_2(x, y))$ accounts for the convection, and $\alpha_1, \alpha_2, \beta$ are coefficients possibly also depending on the space variables. Dirichlet conditions are considered throughout the whole domain boundaries.

Commonly employed discretization methods such as finite elements lead to large-scale sparse linear systems to be solved, typically by means of preconditioned Krylov subspace methods (see, e.g., [18, 28, 32]). Recently, the use of tensor basis discretizations such as finite differences (FD), spectral methods and Isogeometric Analysis (ISA) (see, e.g., [6, 29, 33]) has shown that under certain hypotheses on the domain and coefficients, it is possible to write the discretized version of (1.1) in terms of linear *matrix* equations; in addition to a better understanding of the structural properties of the discretized quantities, this perspective can allow significant savings not only in memory requirements, but also in computational costs. In particular, the FD approach presented in [29] focussed on the case of rectangular domains.

Our aim is to significantly expand the matrix-oriented FD discretization approach by addressing the case where $\mathscr{P}$ is a more general domain, such as a polygon. In our context, this can be addressed by using structured grids. A large range of computational strategies to associate the physical domain with a simpler reference domain such as rectangles or disks have been proposed in the literature. Classically, bilinear projectors have been used in case of curved boundaries and are at the basis of *transfinite interpolation* in two and three dimensions (see, e.g., [19, 25]). Nonetheless it is recognized that, at least from a mathematical standpoint, it may be advantageous to embed the problem in the complex plane and use *conformal and quasi-conformal map-*

Yue Hao, School of Mathematics and Statistics, Lanzhou University, China.
***Corresponding author: Valeria Simoncini,** AM2 and Dipartimento di Matematica, Alma Mater Studiorum – Università di Bologna, Italia, and IMATI-CNR, Pavia, Italia. Email: valeria.simoncini@unibo.it

pings, although their use is strictly constrained to two-dimensional problems[1]. Other explored approaches include variational and other PDE based methods, especially elliptic grid generators, extensively employed in fluid and air dynamics in the 1980s. The literature is extremely vast, here we give the reader a couple of pointers [7, 25, 38]. In this paper we probe the applicability of conformal Schwarz–Christoffel (SC) mappings, which have several theoretical and computational advantages, not least the fact that quite reliable easy-to-use Matlab software is available [13, 14]; other available software leading to the same matrix setting could effectively be employed, such as that described in the Chebfun environment[2]; see also [41].

Our aim is to provide a proof of concept methodology that leads to new algebraic formulations for the differential problem in (1.1) defined in a polygonal domain. To this end, we propose a general use of the Schwarz–Christoffel mappings, that goes beyond the standard Laplacian, which is the most natural operator for Schwarz–Christoffel mappings [15, Ch. 5]. To the best of our knowledge, this application of SC mappings is new.

We take advantage of the recently developed matrix equation framework for solving PDEs discretized using tensor bases [35], and bridge the gap between these bases and the treatment of more general, convex domains for which alternative discretization strategies are commonly employed. Our derivation discusses in detail some of the building blocks, the most crucial one being that the Jacobian determinant of the SC mapping depends on the reference domain variables, giving rise to more complex computations involving Hadamard products, if the tensor basis framework is to be exploited. We discuss how numerical linear algebra techniques need to adapt to the considered operator, while pinpointing both memory and computational properties. In particular, both vector and matrix oriented equations are presented, and their pros and cons outlined. In our numerical experience, it is possible to take advantage at various stages of the properties of the involved operators, as opposed to other discretization strategies such as finite elements, where the differential operator and grid contributions cannot be distinguished.

The outline of the paper is as follows. After a brief introduction of the Schwarz–Christoffel conformal mapping in Section 2, we explore the application of this mapping to the numerical discretization of PDEs in polygonal domains in Section 3, which leads to a linear matrix equation framework. Then several specific examples of classes of PDEs are given in Sections 4 and 5 to illustrate the derivation of the algebraic formulation. In Section 6, various algebraic strategies are considered for solving the resulting matrix equation, and numerical experiments are discussed in Section 7 to show the performance of the discretization procedures and of the adopted solvers. We also analyze the selection of the grids in Section 8. Finally, conclusions of this work are given in Section 9.

All numerical computations throughout the paper were performed in Matlab v.2019a [27] on a Dell laptop using Intel Core i7-3687U with 4 CPUs at 2.10GHz.

2 Discretization and coordinate transformations

Discretization strategies usually aim at transforming a physical region associated with a possibly complex geometry to a simple region through automatic transformations; these transformations can be constructed at the global level (such as in spectral methods) or at the local level (IGA, finite elements). These mappings allow one to then state the problem as an algebraic equation, whose solution approximates the given problem's continuous solution in the original (physical) domain.

A large class of two-dimensional regions can be parameterized by means of special coordinate systems. This is the case for a circle and more generally, for a sector or an annulus, for which computational boundary-conforming grids such as polar coordinates, log-polar coordinates, parabolic cylinder coordinates, elliptic cylinder coordinates and bipolar coordinates transformations can be employed. It should be noticed, how-

1 As it will be clear in the following, the numerics will all be performed in real arithmetic.

2 `https://www.chebfun.org/examples/complex/ConformalMapping.html`

ever, that if either the region or the boundary parameterization is changed, even slightly, these coordinates system may become useless.

As an illustrative example we consider the Poisson equation $-\Delta u = f$ with zero boundary conditions, defined (in Cartesian coordinates) in the annulus sector $\mathscr{P} = \{(x, y) : x = r\cos\vartheta, y = r\sin\vartheta, r \in [r^{(0)}, r^{(1)}], 0 \leqslant r^{(0)} < r^{(1)}, \vartheta \in [0, \pi/4]\}$. Using polar coordinates (r, ϑ), the Poisson equation is given by

$$-r^2 u_{rr} - u_{\vartheta\vartheta} - r u_r = r^2 f(r, \vartheta), \qquad (r, \vartheta) \in [r^{(0)}, r^{(1)}] \times \left[0, \frac{\pi}{4}\right].$$

The discretization of this equation in $(r^{(0)}, r^{(1)})\times(0, \pi/4)$ by FD gives the linear matrix equation (see, e.g., [29]):

$$\Phi^2 TU + UT - \Phi BU = \widetilde{F}.$$

The entry $U_{i,j}$ of the matrix U approximates u at the grid node (r_i, ϑ_j), where r_i's and ϑ_j's are the interior discretization nodes of the intervals $[r^{(0)}, r^{(1)}]$ and $[0, \pi/4]$, respectively. The matrices $B, T \in \mathbb{R}^{n\times n}$ account for the discretization coefficients of the first and second order one-dimensional derivatives, respectively (see Section 3.1 for additional details). Finally, $\Phi = \mathrm{diag}(r_1, r_2, \ldots, r_n)$. Here and in the following, we denote with $\mathrm{diag}(d)$ the diagonal matrix having the components of the vector d on the diagonal.

Although a variety of special coordinate systems are available (see, e.g., [25, Ch. 1]) that can lead to the use of tensor-product based domains, this is not so for simple domains such as polygons with more than four edges. Moreover, it is important to have a computational procedure that automatically transforms a given two-dimensional domain into a rectangle. In the following we explore the use of the Schwarz–Christoffel mapping, a convenient conformal mapping whose computational realization is available via the Matlab software package SC [13, 14].

Other transformations could be considered to arrive at our algebraic framework, as long as the physical domain can be brought into a rectangular computational domain, on which the transformed problem can be solved; we refer the reader to, e.g., [25] for a broad discussion of mappings associated with grid generations.

2.1 Schwarz–Christoffel mapping

Conformal mappings provide a mathematically robust tool for handling complex two-dimensional geometries, also allowing for an easy representation of the physical boundary conditions. From a numerical viewpoint, the resulting grids have the desirable property of being orthogonal, i.e., the tangents to the coordinate curves are perpendicular, which leads to reduced truncation errors when dealing with the transformed differential equations. Moreover, the Jacobian is positive definite by construction, so that the mapping is guaranteed to be well defined (one-to-one), thus preserves the type of partial differential equation after transformation. A conformal mapping can be constructed algebraically or by solving either an integral equation [15, 20, 23, 37, 39], or (systems of) partial differential equations [8, 9, 26]. The Schwarz–Christoffel mapping is one such operator, expressed as an integral [7, 15], e.g., mapping the complex upper half-plane into the interior of a polygon. The 'polygon' may in fact be a quite general region, and may have cracks or vertices at infinity.

A typical example of the Schwarz–Christoffel (SC) map is given by the following function $\mathfrak{g}$, which maps from the upper half of the complex plane to the interior of a polygon. Let the polygon vertices be denoted by $z_1, \ldots, z_n$, and let the numbers $\varphi_1\pi, \ldots, \varphi_n\pi$ be the interior angles at the vertices. The pre-images of the vertices, or pre-vertices, are real and denoted by $\omega_1, \ldots, \omega_n$, and satisfy $\omega_1 < \omega_2 < \cdots < \omega_n = \infty$. The map $\mathfrak{g}$ is defined as

$$\mathfrak{g}(\omega) = \mathfrak{g}(\omega_0) + c\int_{\omega_0}^{\omega} \prod_{j=1}^{n-1} (\zeta - \omega_j)^{\varphi_j - 1}\, \mathrm{d}\zeta \tag{2.1}$$

where ω_0 is a point contained in the upper half of the complex plane, and c is a complex constant. This is known as the Schwarz–Christoffel formula [15, Th. 1.1].

There are two main difficulties associated with computing conformal maps from the Schwarz–Christoffel formula: (i) determining the pre-vertices ω_j, which is the first step for any SC mapping (the so-called SC parameter problem), and (ii) integrating the right-hand side of (2.1). Except for special cases, the pre-vertices ω_j

cannot be determined in closed form; solving the system of equations for the pre-vertices can be a computationally expensive process. As of (ii), finding an explicit expression by evaluating the integral analytically can significantly reduce the time needed to compute the map. Historically, these problems have been difficult to implement numerically, and first successful attempts have been reported in the early 1980s [23, 40]. In the past few years the Schwarz–Christoffel Toolbox for Matlab [13, 14] has arisen as a numerical reliable tool for computationally handling these mappings; the toolbox can solve the parameter problem for various typical domains, and obtain the forward and inverse mappings. That provides us with a powerful and effective tool to numerically solve partial differential equations in a polygonal domain by the FD method, leading to matrix equations. Other software packages associated with conformal or other mappings could also be considered, whenever available (see, e.g., [25, 41]).

3 Schwarz–Christoffel mappings for PDEs in polygonal domains

The Schwarz–Christoffel conformal function $\mathfrak{h}$ mapping the interior of the rectangular domain $\Pi = (a_1, a_2)\times(b_1, b_2)$ (canonical domain) onto the physical domain $\mathscr{P}$ can be obtained as the function $\mathfrak{h}(\omega) = \mathfrak{g}(\mathfrak{l}(\omega))$, where $\mathfrak{l}$ maps the rectangle to the upper half-plane, obtained as the Jacobi elliptic function $\mathrm{sn}(z|m)$, while $\mathfrak{g}$ maps the upper half-plane to the physical polygon $\mathscr{P}$, as before. The functions are related by the chain-rule relation $\mathfrak{h}'(\omega)/\mathfrak{l}'(\omega) = C\prod_{k=1}^{n}(\mathfrak{l}(\omega) - \mathfrak{l}(\omega_k))^{\varphi_k - 1}$ with C and ω_k to be determined [15, formula (4.1)]. Computational and stability arguments lead to the use of an additional intermediate mapping passing through a strip; more details on the actual implementation can be found in [15, Sect. 4.3], and [22]. The rectangle vertices are obtained as pre-images of the polygon vertices chosen as reference (control) points to construct the map; see Section 7.

The cost of using such transformation is an increase in the complexity of the transformed equations, but the domain becomes much simpler, the equations and the boundary conditions become easier to be approximated accurately by the FD method.

Let $(x, y) \in \mathscr{P}$ and $(\xi, \eta) \in \Pi$. Identifying these two sets with corresponding regions in the complex plane, there holds

$$z = \mathfrak{h}(\omega) = \mathfrak{h}(\xi + \mathrm{i}\eta) = x(\xi, \eta) + \mathrm{i}y(\xi, \eta) \tag{3.1}$$

where $x = x(\xi, \eta)$ and $y = y(\xi, \eta)$ are real valued functions. Since $\mathfrak{h}$ is a conformal mapping, the Cauchy–Riemann equations hold, that is

$$x_\xi = y_\eta, \qquad x_\eta = -y_\xi. \tag{3.2}$$

The functions $u(x, y)$ and $f(x, y)$ in $\mathscr{P}$ are transformed to functions $\widetilde{u}$ and $\widetilde{f}$ in Π using

$$\widetilde{u} = \widetilde{u}(\xi, \eta) = u(x(\xi, \eta), y(\xi, \eta)), \qquad \widetilde{f} = \widetilde{f}(\xi, \eta) = f(x(\xi, \eta), y(\xi, \eta))$$

and similarly for $\alpha_i(x, y)$, w_i $(i = 1, 2)$, and $\beta(x, y)$. Without ambiguity, we use $\widetilde{u}$, $\widetilde{f}$, $\widetilde{\alpha}_i$, $\widetilde{w}_i$, and $\widetilde{\beta}$ to denote these transformed functions in the sequel.

The Jacobian matrix of the transformation $\mathfrak{h}$ is

$$\mathcal{J} = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}.$$

Thanks to (3.2) its determinant satisfies

$$\mathscr{J} = \mathscr{J}(\xi, \eta) = x_\xi y_\eta - x_\eta y_\xi = x_\xi^2 + x_\eta^2 > 0.$$

Therefore, the relations $\widetilde{u}_\xi = u_x x_\xi + u_y y_\xi$ and $\widetilde{u}_\eta = u_x x_\eta + u_y y_\eta$ yield

$$\begin{aligned} u_x &= \frac{1}{\mathscr{J}}(\widetilde{u}_\xi y_\eta - \widetilde{u}_\eta y_\xi) = \frac{1}{\mathscr{J}}\left[(\widetilde{u}y_\eta)_\xi - (\widetilde{u}y_\xi)_\eta\right] \\ u_y &= \frac{1}{\mathscr{J}}(\widetilde{u}_\eta x_\xi - \widetilde{u}_\xi x_\eta) = \frac{1}{\mathscr{J}}\left[(\widetilde{u}x_\xi)_\eta - (\widetilde{u}x_\eta)_\xi\right]. \end{aligned} \tag{3.3}$$

Combining with (3.2) gives

$$\begin{aligned} u_{xx} &= \frac{1}{\mathscr{J}}\left[\left(\frac{\widetilde{u}_\xi x_\xi^2 + \widetilde{u}_\eta x_\xi x_\eta}{\mathscr{J}}\right)_\xi + \left(\frac{\widetilde{u}_\eta x_\eta^2 + \widetilde{u}_\xi x_\xi x_\eta}{\mathscr{J}}\right)_\eta\right] \\ u_{yy} &= \frac{1}{\mathscr{J}}\left[\left(\frac{\widetilde{u}_\xi x_\eta^2 - \widetilde{u}_\eta x_\xi x_\eta}{\mathscr{J}}\right)_\xi + \left(\frac{\widetilde{u}_\eta x_\xi^2 - \widetilde{u}_\xi x_\xi x_\eta}{\mathscr{J}}\right)_\eta\right]. \end{aligned} \tag{3.4}$$

Using the formulas (3.2)–(3.4) and after simple calculations, the partial differential equation (1.1) in the physical domain $\mathscr{P}$ can be transformed into the following equation in the canonical domain Π,

$$\begin{aligned} &-(\widetilde{\alpha}_1 x_\xi^2 + \widetilde{\alpha}_2 x_\eta^2)\widetilde{u}_{\xi\xi} - (\widetilde{\alpha}_1 x_\eta^2 + \widetilde{\alpha}_2 x_\xi^2)\widetilde{u}_{\eta\eta} - 2(\widetilde{\alpha}_1 - \widetilde{\alpha}_2)x_\xi x_\eta \widetilde{u}_{\xi\eta} \\ &\quad + \mathscr{J}\left[(\widetilde{w}_1 x_\xi - \widetilde{w}_2 x_\eta) - \frac{1}{\mathscr{J}^2}(\widetilde{\alpha}_1 - \widetilde{\alpha}_2)(3x_\xi x_\eta^2 x_{\xi\xi} + x_\eta^3 x_{\xi\eta} - x_\xi^3 x_{\xi\xi} - 3x_\xi^2 x_\eta x_{\xi\eta})\right]\widetilde{u}_\xi \\ &\quad + \mathscr{J}\left[(\widetilde{w}_1 x_\eta + \widetilde{w}_2 x_\xi) - \frac{1}{\mathscr{J}^2}(\widetilde{\alpha}_1 - \widetilde{\alpha}_2)(x_\eta^3 x_{\xi\xi} + x_\xi^3 x_{\xi\eta} - 3x_\xi x_\eta^2 x_{\xi\eta} - 3x_\xi^2 x_\eta x_{\xi\xi})\right]\widetilde{u}_\eta \\ &\quad + \mathscr{J}^2\widetilde{\beta}\,\widetilde{u} = \mathscr{J}^2\widetilde{f}, \quad (\xi,\eta) \in \Pi \end{aligned} \tag{3.5}$$

or, with obvious notation,

$$-\gamma_1\widetilde{u}_{\xi\xi} - \gamma_2\widetilde{u}_{\eta\eta} + \gamma_3\widetilde{u}_{\xi\eta} + \gamma_4\widetilde{u}_\xi + \gamma_5\widetilde{u}_\eta + \gamma_6\widetilde{u} = \widehat{f}, \qquad (\xi,\eta) \in \Pi. \tag{3.6}$$

The transformed equation thus includes mixed second-order derivatives, in addition to the first and second order derivatives of the original problem. Clearly, the problem simplifies if, for instance, $\widetilde{\alpha}_1 = \widetilde{\alpha}_2$. In particular, for $\widetilde{\alpha}_1 = \widetilde{\alpha}_2 = 1$ from (3.5) we obtain

$$-\mathscr{J}\widetilde{u}_{\xi\xi} - \mathscr{J}\widetilde{u}_{\eta\eta} + \mathscr{J}(\widetilde{w}_1 x_\xi - \widetilde{w}_2 x_\eta)\widetilde{u}_\xi + \mathscr{J}(\widetilde{w}_1 x_\eta + \widetilde{w}_2 x_\xi)\widetilde{u}_\eta + \mathscr{J}^2\widetilde{\beta}\,\widetilde{u} = \mathscr{J}^2\widetilde{f}$$

where a factor $\mathscr{J}$ can be eliminated throughout and the vector $\widetilde{\mathbf{w}}$ highlighted, so that

$$-\widetilde{u}_{\xi\xi} - \widetilde{u}_{\eta\eta} - \widetilde{\mathbf{w}}^T\nabla\widetilde{u} + \mathscr{J}\widetilde{\beta}\,\widetilde{u} = \mathscr{J}\widetilde{f}, \qquad \nabla\widetilde{u} = (\widetilde{u}_\xi, \widetilde{u}_\eta)^T. \tag{3.7}$$

This simplified form stresses the role of the mapping on the various terms, elegantly and clearly discerning between the operator and domain discretization parts of the numerical procedure.

3.1 Discretization of the transformed PDE

To approximate the transformed equation (3.6) by FDs, we define the grid points $(\xi_i, \eta_j) \in \Pi = (a_1, a_2) \times (b_1, b_2)$ by

$$\begin{aligned} h_1 &= \frac{a_2 - a_1}{n_1 + 1}, \qquad \xi_i = a_1 + ih_1, \qquad 1 \leqslant i \leqslant n_1 \\ h_2 &= \frac{b_2 - b_1}{n_2 + 1}, \qquad \eta_j = b_1 + jh_2, \qquad 1 \leqslant j \leqslant n_2 \end{aligned}$$

where n_1 and n_2 are the number of interior grid points in the intervals (a_1, a_2) and (b_1, b_2), respectively. To avoid a pedantic notation, with some abuse of notation in the following we shall not distinguish between n_i and $n_i + 2$, which corresponds to either include or exclude the interval boundary points.

The transformation $x = x(\xi, \eta)$ and $y = y(\xi, \eta)$ carries the canonical-space grid to a physical-space grid $(x_{i,j}, y_{i,j})$, where

$$x_{i,j} = x(\xi_i, \eta_j), \qquad y_{i,j} = y(\xi_i, \eta_j), \qquad 1 \leqslant i \leqslant n_1, \qquad 1 \leqslant j \leqslant n_2.$$

One of the convenient features of these conformal mappings is that the derivatives x_η, x_ξ can cheaply and accurately be computed by using the mapping itself [14, Table III]. In matrix notation, these will be denoted by $(X_\eta)_{ij} \approx x_\eta(\xi_i, \eta_j)$ and $(X_\xi)_{ij} \approx x_\xi(\xi_i, \eta_j)$.

The discretization also induces discrete values for all of the transformed variables in (3.6). For example, $\widetilde{u}_{i,j} = \widetilde{u}(\xi_i, \eta_j) = u(x(\xi_i, \eta_j), y(\xi_i, \eta_j)) = u(x_{i,j}, y_{i,j}) = u_{i,j}$.

Let $U_{i,j}$ be the value of the approximation to $u_{i,j}$. In order to attain second-order accuracy of the discretization of the two-dimensional problems (3.6), the following standard FD approximations are adopted for $i = 1, \dots, n_1$, $j = 1, \dots, n_2$,

$$\begin{aligned}
\widetilde{u}_{\xi\xi}(\xi_i, \eta_j) &\approx \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h_1^2} \\
\widetilde{u}_{\xi\eta}(\xi_i, \eta_j) &\approx \frac{U_{i+1,j+1} - U_{i+1,j-1} - U_{i-1,j+1} + U_{i-1,j-1}}{4h_1h_2} \\
\widetilde{u}_{\eta\eta}(\xi_i, \eta_j) &\approx \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h_2^2} \\
\widetilde{u}_{\xi}(\xi_i, \eta_j) &\approx \frac{U_{i+1,j} - U_{i-1,j}}{2h_1} \\
\widetilde{u}_{\eta}(\xi_i, \eta_j) &\approx \frac{U_{i,j+1} - U_{i,j-1}}{2h_2}.
\end{aligned} \tag{3.8}$$

These approximations, performed element-wise in the grid, allow one to derive linear matrix equations in the unknown matrix $U_{i,j}$, directly associated with the grid points (ξ_i, η_j), and thus with the corresponding nodes $(x_{i,j}, y_{i,j})$ in the physical domain. In particular, by collecting the coefficients in the linear combinations of adjacent nodes, we see that (see, e.g., [29]):

$$\widetilde{u}_{\xi\xi}(\xi_i, \eta_j) \approx (T_1 U)_{i,j}, \qquad T_1 = \frac{1}{h_1^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n_1 \times n_1} \tag{3.9}$$

and

$$\widetilde{u}_{\eta\eta}(\xi_i, \eta_j) \approx (UT_2)_{i,j}, \qquad T_2 = \frac{1}{h_2^2}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n_2 \times n_2}. \tag{3.10}$$

A more detailed description of the other terms will be given in later sections.

In the following, several specific examples of classes of PDEs are given to illustrate the derivation of the actual matrix equation formulation.

4 Poisson equation

The simplest possible case to be considered is the following Poisson equation

$$-u_{xx} - u_{yy} = f, \qquad (x, y) \in \mathscr{P} \tag{4.1}$$

with Dirichlet boundary conditions. The Laplace operator is the most representative of the strength of Schwarz–Christoffel transformations, as the operator is invariant under conformal mappings (see, e.g., [2, 8, 15]). Hence, according to (3.7), the transformed equation associated with (4.1) is

$$-\widetilde{u}_{\xi\xi} - \widetilde{u}_{\eta\eta} = \mathscr{J}\widetilde{f}, \qquad (\xi, \eta) \in \Pi. \tag{4.2}$$

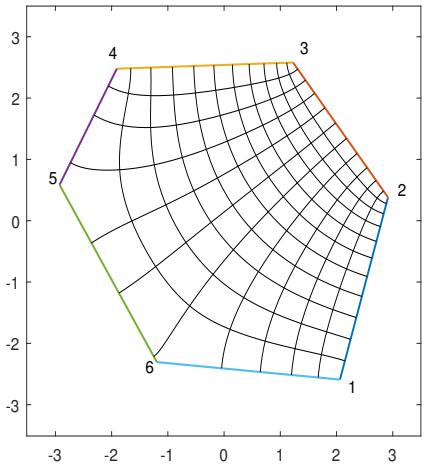

Fig. 1: Convex domain. Grid of the Schwarz–Christoffel mapping over the physical domain.

We note that this problem serves as an introduction to more involved equations, since built-in solvers for problems only involving the Laplace operator are already available in the SC Toolbox.

With the FD approximations (3.9)–(3.10), the discretized matrix form of equation (4.2) is given by

$$T_1 U + U T_2 = F \tag{4.3}$$

where

$$F = \widetilde{F} + \text{b.c.}, \qquad \widetilde{F}_{i,j} = (\mathscr{J}\widetilde{f})(\xi_i, \eta_j), \qquad 1 \leqslant i \leqslant n_1, \quad 1 \leqslant j \leqslant n_2. \tag{4.4}$$

The linear equation (4.3) is a Sylvester matrix equation, and can be solved by the Bartels–Stewart method or its variants [1, 21, 36], or by iterative methods [35], depending on the matrix size. We note that the right-hand side matrix F is computed element-wise as $(\mathscr{J}\widetilde{f})(\xi_i, \eta_j)$. Let J be the matrix accounting for $\mathscr{J}(\xi_i, \eta_j)$ for all interior grid nodes. The occurrence of element-wise computations with J provides an important challenge in the solution of the matrix equation form, which will become apparent when solving more involved equations in the next sections.

Let $\mathscr{P}$ be the interior of the hexagon in Fig. 1 having vertices[3] (which are ordered counter-clock-wise)

$$z_1 = (2.06107, -2.59033), \quad z_2 = (2.91603, 0.38168), \quad z_3 = (1.22646, 2.58015)$$

$$z_4 = (-1.90840, 2.47837), \quad z_5 = (-2.92621, 0.58524), \quad z_6 = (-1.19593, -2.30534).$$

Using the SC Matlab Toolbox [13, 14], the obtained canonical (rectangular) domain is given by rectangle $\Pi = [-1.60167736, 1.60167736] \times [0, 2.71248244]$. A sample of the graphical representation of this grid mapped by $\mathfrak{h}$ to $\mathscr{P}$ is also shown in Fig. 1, while a finer grid (with $n_1 = n_2 = 30$ grid points in each direction in Π) is used in our experiment. We solve the problem using two different settings: we first use homogeneous (zero) Dirichlet boundary conditions, and then, denoting $z_i = (x_i, y_i)$, we use the following boundary conditions:

$$\begin{aligned} &u = (x - x_1)(x - x_3) + (y - y_1)(y - y_3) + 1 \quad \text{on edges } z_1 - z_2 \text{ and } z_2 - z_3 \\ &u = 1 \quad \text{on other edges} \end{aligned} \tag{4.5}$$

ensuring the solution continuity at the boundary.

The approximate solution U to (4.1) for $f = 1$ so that $F = J$ under zero boundary conditions is displayed in Fig. 2 (left), while that associated with the non-uniform boundary conditions is reported in Fig. 2 (right). Due to the small problem size, the computational results were obtained with the Matlab function `lyap` [27].

3 The polygon was obtained by hand using the SC Toolbox `polyedit` graphical editor (see Section 7). This explains the seemingly peculiar choice of the vertices $\{z_i\}$, whose first 6 significant digits are reported here.

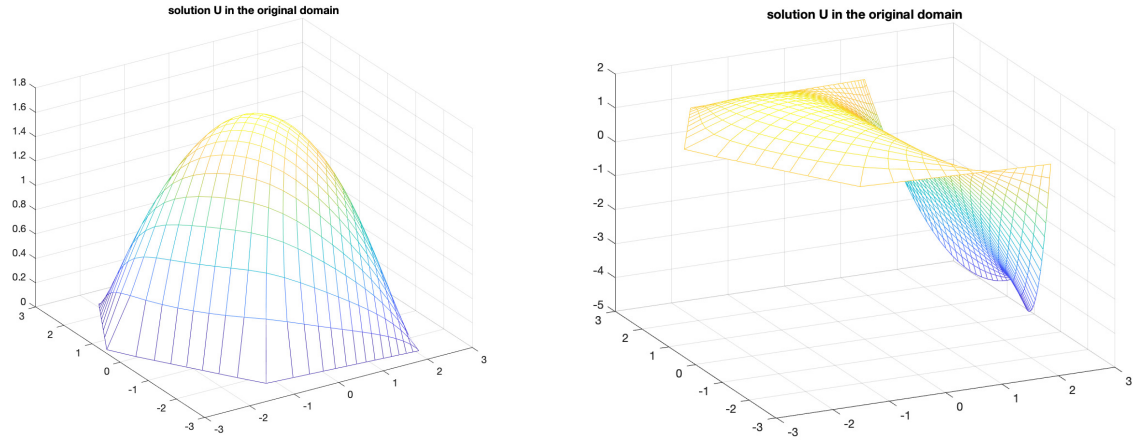

Fig. 2: Poisson equation solution U in the physical domain $\mathscr{P}$ with homogeneous (left) and non-constant (right) Dirichlet boundary conditions.

5 Beyond the Laplace operator

In (1.1), the presence of space-dependent coefficients α_1, α_2, $\mathbf{w}$, and β provides an important challenge to the discretized matrix equation setting. If $\mathscr{P}$ were a rectangle, non-constant but separable coefficients would be easy to deal with. For the sake of the discussion, let us consider the zero-order term. Assuming for instance that $\beta(x, y) = \beta_1(x)\beta_2(y)$, then the term βu would be discretized as $\boldsymbol{\beta}_1 U \boldsymbol{\beta}_2$, where $\boldsymbol{\beta}_1 = \mathrm{diag}(\beta_1(x_1), \ldots, \beta_1(x_{n_1}))$ and analogously for $\boldsymbol{\beta}_2$ (see, e.g., [29]). In the case of a polygon, equation (3.7) shows that the term βu is replaced by $\mathscr{J}\tilde{\beta}u$, where all quantities depend on the space variables, thus the product needs to be carried out element by element, even in the case that β is separable, since $\tilde{\beta}$ and $\mathscr{J}$ are not separable functions in general.

If G denotes the matrix associated with the discretization of $\mathscr{J}\tilde{\beta}$, then the term $\mathscr{J}\tilde{\beta}u$ leads to a matrix term in the matrix equation of the form $G \circ U$, where '$\circ$' denotes the (element-wise) Hadamard product. It should be noticed, however, that if G is a low rank matrix, that is it satisfies $G = ZQ^T$ with $Z = [z_1, \ldots, z_\ell]$, $Q = [q_1, \ldots, q_\ell]$ and $\ell \ll \min\{n_1, n_2\}$, then thanks to the properties of the Hadamard product, we could write $G \circ U = \mathrm{diag}(z_1)U\mathrm{diag}(q_1) + \ldots + \mathrm{diag}(z_\ell)U\mathrm{diag}(q_\ell)$, Hence, in the case G can be well approximated by a low rank matrix, the Hadamard product can be substituted with a sum of matrix terms in the usual matrix product.

In the next sections we dwell with the algebraic formulations for a few classes of differential problems.

5.1 The Poisson equation with a reaction term

Consider the numerical solution of

$$\begin{cases} -\Delta u + \beta u = f, & (x, y) \in \mathscr{P} \\ u = 0, & (x, y) \in \partial\mathscr{P}. \end{cases} \tag{5.1}$$

The addition of the quantity βu, with $\beta = \beta(x, y)$, increases the difficulty of the algebraic setting, since it adds a composite term to the matrix equation. Using (3.7) we can write

$$\begin{cases} -\tilde{u}_{\xi\xi} - \tilde{u}_{\eta\eta} + (\mathscr{J}\tilde{\beta})\tilde{u} = \mathscr{J}\tilde{f}, & (\xi, \eta) \in \Pi \\ \tilde{u} = 0, & (\xi, \eta) \in \partial\Pi. \end{cases} \tag{5.2}$$

Proceeding with the FD discretization, we denote[4]

$$G_6(i,j) := (\mathscr{J}\tilde{\beta})(\xi_i,\eta_j), \qquad 1 \leqslant i \leqslant n_1, \quad 1 \leqslant j \leqslant n_2 \tag{5.3}$$

thus giving the following matrix equation, corresponding to equation (5.2),

$$T_1 U + U T_2 + G_6 \circ U = F \tag{5.4}$$

where T_1, T_2 and U, F are defined in Section 3.1.

5.2 Variable coefficient diffusion equation

Consider the variable coefficient equation

$$\begin{cases} -\alpha_1 u_{xx} - \alpha_2 u_{yy} = f, & (x,y) \in \mathscr{P} \\ u = 0, & (x,y) \in \partial\mathscr{P} \end{cases} \tag{5.5}$$

with $\alpha_i = \alpha_i(x,y)$, $i = 1, 2$. According to (3.6), the transformed equation is

$$\begin{cases} -\gamma_1 \tilde{u}_{\xi\xi} - \gamma_2 \tilde{u}_{\eta\eta} + \gamma_3 \tilde{u}_{\xi\eta} + \gamma_4 \tilde{u}_{\xi} + \gamma_5 \tilde{u}_{\eta} = \widehat{f}, & (\xi,\eta) \in \Pi \\ \tilde{u} = 0, & (\xi,\eta) \in \partial\Pi \end{cases} \tag{5.6}$$

with

$$\begin{aligned} \gamma_1 &= \tilde{\alpha}_1 x_\xi^2 + \tilde{\alpha}_2 x_\eta^2, \quad \gamma_2 = \tilde{\alpha}_1 x_\eta^2 + \tilde{\alpha}_2 x_\xi^2, \quad \gamma_3 = 2(\tilde{\alpha}_2 - \tilde{\alpha}_1) x_\xi x_\eta \\ \gamma_4 &= \frac{1}{\mathscr{J}}(\tilde{\alpha}_2 - \tilde{\alpha}_1)(3x_\xi x_\eta^2 x_{\xi\xi} + x_\eta^3 x_{\xi\eta} - x_\xi^3 x_{\xi\xi} - 3x_\xi^2 x_\eta x_{\xi\eta}) \\ \gamma_5 &= \frac{1}{\mathscr{J}}(\tilde{\alpha}_2 - \tilde{\alpha}_1)(x_\eta^3 x_{\xi\xi} + x_\xi^3 x_{\xi\eta} - 3x_\xi x_\eta^2 x_{\xi\eta} - 3x_\xi^2 x_\eta x_{\xi\xi}) \end{aligned} \tag{5.7}$$

and $\widehat{f} = \mathscr{J}^2 \tilde{f}$. Given the $\zeta \times \zeta$ matrix

$$B = \begin{bmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 0 \end{bmatrix} \tag{5.8}$$

we denote by

$$\begin{aligned} B_1 &:= \frac{1}{2h_1} B \in \mathbb{R}^{n_1 \times n_1}, \quad \zeta = n_1 \\ B_2 &:= \frac{1}{2h_2} B^T \in \mathbb{R}^{n_2 \times n_2}, \quad \zeta = n_2 \end{aligned}$$

the matrices associated with the discretization of the first order derivative in ξ and η, respectively, and for $\ell = 1, \ldots, 5$ we set

$$(G_\ell)_{i,j} = \gamma_\ell(\xi_i, \eta_j), \qquad i = 1, \ldots, n_1, \quad j = 1, \ldots, n_2. \tag{5.9}$$

With these notations, we get the following matrix equation for the problem (5.6):

$$G_1 \circ (T_1 U) + G_2 \circ (U T_2) + G_3 \circ (B_1 U B_2) + G_4 \circ (B_1 U) + G_5 \circ (U B_2) = F \tag{5.10}$$

where T_1, T_2 and U, F are defined in Section 3.1.

4 The subscript in G_6 accounts for more terms that will arise in the following, associated with the discretization of higher order terms.

5.3 Convection–diffusion equation

Consider the numerical solution of the partial differential equation

$$\begin{cases} -\varepsilon\Delta u + u_x = f, & (x,y) \in \mathscr{P} \\ u = 0, & (x,y) \in \partial\mathscr{P}. \end{cases} \tag{5.11}$$

According to (3.7) the corresponding transformed equation is

$$\begin{cases} -\varepsilon(\widetilde{u}_{\xi\xi} + \widetilde{u}_{\eta\eta}) + x_\xi \widetilde{u}_\xi + x_\eta \widetilde{u}_\eta = \mathscr{J}\widetilde{f}, & (\xi,\eta) \in \Pi \\ \widetilde{u} = 0, & (\xi,\eta) \in \partial\Pi. \end{cases} \tag{5.12}$$

Using the approximations in (3.8) gives the following matrix equation

$$\varepsilon(T_1 U + U T_2) + X_\xi \circ (B_1 U) + X_\eta \circ (U B_2) = F \tag{5.13}$$

where T_1, T_2, U, F and B_1, B_2 are defined in Section 3.1 and after (5.8), respectively, while we recall that

$$(X_\xi)_{i,j} = x_\xi(\xi_i, \eta_j), \quad (X_\eta)_{i,j} = x_\eta(\xi_i, \eta_j), \qquad i = 1, \dots, n_1, \quad j = 1, \dots, n_2.$$

6 Numerical solution of the matrix equation

Discretization of the transformed equation (3.5) by standard FDs leads to a medium scale linear matrix equation, This may be considered the major advantage over vector oriented discretizations. In fact, if the original PDE were to be solved by a method leading to a vector-based algebraic equation, one would be then faced with the solution of a linear system of size of the order of $n_1 n_2$, for a comparable discretization accuracy. In that case too, a direct solution would be unfeasible. On the other hand, the presence of coefficient matrices involving the Hadamard product may make the computations more cumbersome.

The complexity of the solution strategy and its computational costs for solving discretized PDEs in the form (1.1) usually depend on whether the differential operator is diffusion dominated or convection dominated. In our matrix oriented context, we experienced a similar situation, therefore more than one method may be considered.

6.1 Fixed point iteration

Let us write the left-hand side of matrix equation (5.10) as the sum of two matrix terms, so that

$$\mathcal{L}(U) - \mathcal{N}(U) = F$$

where the choice of $\mathcal{L}$ depends on the actual continuous problem. Assuming $\mathcal{L}$ to be an invertible operator, and given a starting approximation U_0, a sequence of approximates $\{U_k\}$ can be obtained by a stationary iteration

$$\text{Solve} \quad \mathcal{L}(U_{k+1}) = F + \mathcal{N}(U_k), \qquad k = 0, 1, \dots$$

or, alternatively, $U_{k+1} = U_k + Z_k$, where Z_k solves $\mathcal{L}(Z_k) = R_k$ and $R_k = F - \mathcal{L}(U_k) + \mathcal{N}(U_k)$ is the current residual matrix. This approach is classical in the matrix equation context, and has led to ADI-type iterations already in the 1960s [17, 42]. Since then, fixed point iterations and splitting approaches have led to a rich literature for linear and nonlinear matrix equations with two or more terms, possibly under particular hypotheses, to ensure that the iteration is convergent and computationally feasible (see, e.g., [3, 4, 10–12, 24, 34]).

The effectiveness of the approach depends on the cost of solving with the chosen operator $\mathcal{L}$, and on the number of iterations needed to reach the required accuracy, which in turn depends on the spectral properties of the iteration operator, that is of $\mathcal{L}^{-1}(\mathcal{N}(\cdot))$. In general, we found it hard to get convergence of this approach without any a priori spectral knowledge of the operators involved.

6.2 Vector iterative solver with matrix-oriented preconditioning

A typically effective alternative to stationary iterations is the use of standard (vector) iterative solvers. The discretization by FDs (or other methods) of (3.6) in the rectangle Π classically leads to a system of the form (see, e.g., [28, 32]):

$$\mathcal{A}\mathbf{u} = \mathbf{f} \tag{6.1}$$

where the components of $\mathbf{u}$ collect the entries of U_{ij} in lexicographic order, while $\mathcal{A}$ accounts for the discretization of all derivatives in the two space variables. In our context, we can derive the vector form by using the Kronecker operator, defined as (see, e.g., [21, Sect. 1.3.6]):

$$A \otimes B = \begin{bmatrix} A_{1,1}B & \cdots & A_{1,n}B \\ \vdots & \ddots & \vdots \\ A_{n,1}B & \cdots & A_{n,n}B \end{bmatrix}.$$

This matrix product satisfies

$$\mathrm{vec}(AXB) = (B^T \otimes A)\,\mathrm{vec}(X) \tag{6.2}$$

where $\mathrm{vec}(X)$ stacks the columns of X one below the other. We also recall that $\mathrm{vec}(G \circ M) = \mathrm{diag}(\mathrm{vec}(G))\mathrm{vec}(M)$ for any matrices G, M of equal dimensions. The vectorization of (5.10) is thus given by (6.1) obtained as

$$\big(D_1(I \otimes T_1) + D_2(T_2^T \otimes I) + D_3(B_2^T \otimes B_1) + D_4(I \otimes B_1) + D_5(B_2^T \otimes I)\big)u = f$$

where $D_i = \mathrm{diag}(\mathrm{vec}(G_i))$. Using classical strategies, the system to be solved has very large dimensions, with a sparse $\mathcal{A}$ of size $n_1 n_2 \times n_1 n_2$, so that preconditioned iterative Krylov subspace methods are usually employed, such as CG, MINRES or GMRES [32]. To limit memory consumptions for $\mathcal{A}$ nonsymmetric, a restarted version is usually employed, with a restart occurring every m iterations.

A large variety of preconditioners can be employed, such as incomplete LU, algebraic multigrid, and also operator strategies (see, e.g., [5, 18, 28]). In the last class we can include operators that take into account the matrix equation structure of the problem, resulting from the performed transformation. This last approach has recently been used in algebraic equations stemming from different discretization techniques (see, e.g., [29, 33]). The performance of the iterative scheme thus strongly depends on the effectiveness of the preconditioner. We shall see that operator based preconditioners can be quite robust, and may lead to mesh independent performance, in terms of number of iterations; see, e.g., [28, Ch. 4] for a general discussion on mesh independence of preconditioning strategies.

6.3 Galerkin projection method

An approach that has recently shown its effectiveness in solving linear matrix equations with several terms consists of reducing the problem dimension by projecting the equation onto a smaller space.

To derive the method in its generality, we write the linear matrix equation as

$$G_1 \circ (T_1 U) + G_2 \circ (U T_2) + G_3 \circ (B_1 U B_2) + G_4 \circ (B_1 U) + G_5 \circ (U B_2) + G_6 \circ U = F$$

or, in short, $\mathcal{S}(U) = F$.

Let $\mathcal{V}$ and $\mathcal{W}$ be two subspaces of $\mathbb{R}^n$ (here for simplicity we assume $n_1 = n_2 = n$), and let the k columns of V_k (W_k) be orthonormal bases for $\mathcal{V}$ ($\mathcal{W}$) with $k \ll n$. Then we look for an approximation

$$U^{(k)} = V_k Y_k W_k^T \approx U \tag{6.3}$$

and let

$$R_k := F - \mathcal{S}(U^{(k)}) \tag{6.4}$$

be the associated residual. To determine Y_k, an orthogonality (Galerkin) condition is imposed on the residual matrix. With the matrix inner product, this corresponds to imposing

$$V_k^T R_k W_k = 0 \tag{6.5}$$

that is

$$V_k^T \mathcal{S}(V_k Y_k W_k^T) W_k = V_k^T F W_k.$$

As an example, consider equation (5.4), where $G_1 = G_2 = \mathbf{1}$ (the matrix of all ones) so that $G_1 \circ (T_1 U) = T_1 U$ and analogously for G_2, while all other G_is are zero except G_6. We thus obtain

$$V_k^T T_1 V_k Y_k + Y_k W_k^T T_2 W_k + V_k^T (G_6 \circ (V_k Y_k W_k^T)) W_k = V_k^T F W_k. \tag{6.6}$$

This reduced dimension equation is solved by resorting to its vectorization by means of the Kronecker product in (6.2). Therefore, we can write $u^{(k)} := \mathrm{vec}(U^{(k)}) = (W_k \otimes V_k)\mathrm{vec}(Y_k)$. For the example above, we thus have

$$\mathcal{A}_k y := (I \otimes (V_k^T T_1 V_k))y + ((W_k^T T_2^T W_k) \otimes I)y + (W_k \otimes V_k)^T D_6 (W_k \otimes V_k)y = f_k \tag{6.7}$$

where $D_6 = \mathrm{diag}(\mathrm{vec}(G_6))$ and $f_k = (W_k \otimes V_k)^T \mathrm{vec}(F)$. This system can be solved by either direct or iterative methods. In the absence of ad-hoc effective methods for solving multiterm small size dense matrix equations such as (6.6), in our experiments we used Matlab backslash for $\mathcal{A}_k$. The iterative solution of (6.7) exploiting the matrix-oriented form has been explored in similar contexts though here the coefficients are generally dense. The iteration would lead to a so-called inner-outer scheme (see, e.g., [31] and references therein). However, appropriately designed procedures addressing the generic case (6.6) would be desirable.

To make the whole procedure efficient, the cost of forming the third term in the coefficient matrix in (6.7) should be decreased, as it involves matrices with dimension n vectors. The following results show how this computational cost can be significantly decreased.

Proposition 6.1. Let $M, N \in \mathbb{R}^{n\times k_1}$, $Z, S \in \mathbb{R}^{n\times k_2}$, and let $D \in \mathbb{R}^{n\times n}$ be diagonal. Define the $k_1k_2 \times k_1k_2$ matrix

$$\mathcal{H} := (M \otimes Z)^T \mathrm{diag}(\mathrm{vec}(D))(N \otimes S) = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1k_1} \\ H_{21} & H_{22} & & H_{2k_1} \\ \ddots & \ddots & \ddots & \vdots \\ & \ddots & & H_{k_1k_1} \end{bmatrix}, \qquad H_{ij} \in \mathbb{R}^{k_2\times k_2}. \tag{6.8}$$

Then the (i, j)-block of $\mathcal{H}$ is

$$H_{ij} = Z^T \widehat{D}_{i,j} S, \qquad \widehat{D}_{i,j} = \mathrm{diag}\left(D(M_{:,i} \circ N_{:,j})\right) \tag{6.9}$$

where $M_{:,i}$ ($N_{:,j}$) denotes the ith column of M (the jth column of N).

Proof. Let $\mathcal{D}_i = \mathrm{diag}(D_{:,i}) \in \mathbb{R}^{n\times n}$. Then for the (i, j)-block of $\mathcal{H}$, we have

$$H_{ij} = \sum_{l=1}^{n} M_{l,i} Z^T \mathcal{D}_l N_{l,j} S = Z^T \left(\sum_{l=1}^{n} M_{l,i} \mathcal{D}_l N_{l,j} \right) S = Z^T \mathrm{diag}\big(D\left(M_{:,i} \circ N_{:,j}\right)\big) S. \tag{6.10}$$

This completes the proof. □

The next result describes how to update the matrix $\mathcal{H}$ when the building blocks increase their size.

Proposition 6.2. Assume the notation of Proposition 6.1 holds, and that the columns of the matrices M, N and Z, S are increased by one, i.e., their dimensions are $n \times (k_1 + 1)$ and $n \times (k_2 + 1)$, respectively.
(i) If $i \leqslant k_1$ and $j \leqslant k_1$, then

$$H_{ij} = \begin{bmatrix} Z_{k_2}^T \widehat{D}_{i,j} S_{k_2} & Z_{k_2}^T \widehat{D}_{i,j} S_{:,k_2+1} \\ Z_{:,k_2+1}^T \widehat{D}_{i,j} S_{k_2} & Z_{:,k_2+1}^T \widehat{D}_{i,j} S_{:,k_2+1} \end{bmatrix}$$

where Z_{k_2} collects the first k_2 columns of Z. In particular, the first block $Z_{k_2}^T \widehat{D}_{i,j} S_{k_2}$ is the matrix H_{ij} in the previous step;

(ii) If $i = k_1 + 1$ or $j = k_1 + 1$, then $H_{ij} = Z^T \widehat{D}_{i,j} S$.

The results in (6.1)–(6.2) give a simplified way to compute the quantity $V_k^T(G_6 \circ (V_k Y_k W_k^T))W_k$ and other similar quantities that arise in explicitly computing $V_k^T \mathcal{S}(V_k Y_k W_k^T) W_k$. In particular, using $D_6 = \mathrm{diag}(\mathrm{vec}(G_6))$ we can write the (i, j)-block of $(W_k \otimes V_k)^T D_6 (W_k \otimes V_k)$ as

$$[(W_k \otimes V_k)^T D_6 (W_k \otimes V_k)]_{ij} = V_k^T \mathrm{diag}\left(G_6(W_{:,i} \circ W_{:,j})\right) V_k.$$

Here and in the following we denote by $W_{:,i}$ the ith column of the matrix W_k, omitting the subscript k to lighten the notation.

The following result provides the application of the expressions above.

Proposition 6.3. Partition the matrix $\mathcal{A}_k \in \mathbb{R}^{k^2 \times k^2}$ in the equation (6.7) into $k \times k$ blocks with the size of each block being $k \times k$, then the (i, j)-block of matrix $\mathcal{A}_k$ is $A_{ij}^{(k)} = V_k^T G_{i,j} V_k$, where

$$G_{i,j} := \mathrm{diag}\left(W_{:,i} \circ W_{:,j}\right) T_1 + \mathrm{diag}\left(W_{:,i} \circ (T_2 W_k)_{:,j}\right) + \mathrm{diag}\left(G_6(W_{:,i} \circ W_{:,j})\right).$$

Note that the expression for $A_{ij}^{(k)}$ can be given in a simplified formula as follows:

$$A_{ij}^{(k)} = \delta_{ij} V_k^T T_1 V_k + (W_k^T T_2 W_k)_{i,j} I + V_k^T \mathrm{diag}\left(G_6(W_{:,i} \circ W_{:,j})\right) V_k$$

where δ_{ij} is the Kronecker delta function. Furthermore, when the dimension of the subspaces $\mathcal{V}$ and $\mathcal{W}$ is increased by one, the (i, j)-block of matrix $\mathcal{A}_{k+1}$ is

$$\begin{cases} A_{ij}^{(k+1)} = \begin{bmatrix} A_{ij}^{(k)} & V_k^T G_{i,j} V_{:,k+1} \\ V_{:,k+1}^T G_{i,j} V_k & V_{:,k+1}^T G_{i,j} V_{:,k+1} \end{bmatrix}, & i \leqslant k \text{ and } j \leqslant k; \\ A_{ij}^{(k+1)} = V_{k+1}^T G_{i,j} V_{k+1}, & i = k+1 \text{ or } j = k+1. \end{cases}$$

It was shown in [30] that if the operator associated with the problem is symmetric and positive definite, then the Galerkin projection is optimal, in the sense that it minimizes the error in the corresponding norm. More precisely, the following result holds (see [30]).

Proposition 6.4. Let $\mathcal{S}(X) = F$ with $\mathcal{S} : X \mapsto \sum_{j=1}^{\ell} A_j X B_j$ be a symmetric and positive definite operator[5], with $A_j \in \mathbb{R}^{n_1 \times n_1}$, $B_j \in \mathbb{R}^{n_2 \times n_2}$. Let $U_\star$ be the exact solution to the problem $\mathcal{S}(U) = F$, and let range(V_k), range(W_k) be the constructed approximation spaces, so that $U_k = V_k Y_k W_k^T$ is the Galerkin approximate solution. Then

$$\|U_\star - U_k\|_{\mathcal{S}} = \min_{\substack{Z = V_k Y W_k^T \\ Y \in \mathbb{R}^{k \times k}}} \|U_\star - Z\|_{\mathcal{S}}$$

where the norm is defined as $\|X\|_{\mathcal{S}}^2 = \mathrm{trace}\left(\sum_{j=1}^{\ell} X^T A_j X B_j\right)$.

In our setting the situation may be different whenever Hadamard product terms occur in the operator. Nonetheless, this is not the case for the constant coefficient Poisson equation in (4.1), where indeed error minimization is achieved. This property will be used in the discussion of Section 8.1.

In terms of memory requirements, the advantage of the matrix projection approach over vector methods is that only vectors of length n_1 and n_2 are stored, as opposed to vectors of length $n_1 n_2$. Their number depends on the maximum dimensions of the spaces that need to be generated, and this number is hard to determine a priori. Nonetheless, some intuition can be obtained by a preliminary experiment with a coarse grid, since we have experimental evidence that the space dimensions grow only sublinearly with the number of nodes in each direction. Clearly, the effectiveness of the projection approach also depends on the fact that $k \ll n$, to ensure that handling vectors of length k^2 does not provide a significant overload. The selection of the approximation spaces is thus crucial.

5 The operator $\mathcal{S}$ is symmetric and positive definite if and only if its Kronecker form matrix $\sum_{j=1}^{\ell} B_j^T \otimes A_j$ is.

6.4 Choosing the approximation spaces

The choice of the approximation space is important for the overall effectiveness of the projection. To this end, we consider rational Krylov subspaces, which have been shown to be applicable to the case of multiterm linear matrix equations [35]. In its general form, a rational Krylov space is defined as

$$\mathcal{RK}_k(\mathbb{A}, \mathbb{B}, V_0) = \operatorname{range}\Big(\Big[V_0, (\mathbb{A} + s_2\mathbb{B})^{-1}V_0, \ldots, \prod_{\ell=2}^{k}(\mathbb{A} + s_\ell\mathbb{B})^{-1}V_0\Big]\Big)$$

where V_0 is a tall matrix with $\widehat{\ell}$ linearly independent columns, and $\mathbb{A}$ and $\mathbb{B}$ are matrices of equal size, compatible with those of V_0. The space dimension will thus be not greater than $\widehat{\ell}k$. The parameters s_j can be computed a priori or dynamically during the iteration. We remark that an orthonormal basis for this space is determined by an Arnoldi-type procedure, which allows one to add vectors to the basis in an incremental fashion [35].

For equations with several terms, there is no obvious recipe on which matrices $\mathbb{A}$, $\mathbb{B}$ should be used to build the left and right approximation spaces. The principle is that the generated space should well approximate the spectral properties of all coefficient matrices. To this end, various strategies have been devised to, e.g., transform the coefficient matrices so that they all have eigenvalues in the same spectral interval, possibly of moderate length (see, e.g., [31]). Clearly, available a priori spectral information is crucial for the success of this procedure. In our case, except for the Hadamard product, the coefficient matrices stem from operators of different orders, of which the second order ones may be dominant. We tested different options, and found that the choice $\mathbb{B} = I$ and $\mathbb{A} = L_j^{-1}B_jL_j^{-T}$ was the most appropriate, where $T_j = L_jL_j^T$ is the Cholesky factorization of T_j, $j = 1, 2$; the choice $j = 1$ was made for the space $\mathcal{V}$, and $j = 2$ for $\mathcal{W}$. The presence of the Hadamard product makes any theoretical analysis very hard, since this product does not easily preserve spectral properties. A deeper study would be of great interest, but it is beyond the goals of this paper.

For selecting the shifts s_j we followed the greedy procedure first proposed in [16] in the nonsymmetric case, and used the projected matrices $V_k^T(T_1+B_1)V_k$ and $W_k^T(T_2+B_2)W_k$ for that purpose. We refer the reader to [16, 35] for additional details and a more general picture associated with the actual procedure.

In our context the selection of V_0 for each space was also to be made. Since the right-hand side F is not low rank, the truncated singular value decomposition (SVD) of F was performed, namely $F \approx P_\ell\Sigma_\ell Z_\ell^T$, where P_ℓ, Z_ℓ have ℓ columns each, while $\Sigma_\ell = \operatorname{diag}(\sigma_1, \ldots, \sigma_\ell)$ and the σ_js are the first ℓ singular values of F, in decreasing order. Hence, the first $\widehat{\ell} \leqslant \ell$ columns of P_ℓ (of Z_ℓ) were selected to start building V_k (W_k), where $\widehat{\ell}$ is the first integer such that

$$\frac{\sum_{j=1}^{\widehat{\ell}}\sigma_j}{\sum_{j=1}^{\ell}\sigma_j} > 1 - \texttt{tol} \tag{6.11}$$

where `tol` is the tolerance used in the stopping criterion of the projection method.

The method did not seem to be overly sensitive to the choice of $\widehat{\ell}$: while changing the tolerance in (6.11) changed $\widehat{\ell}$, the final subspace dimensions to reach convergence did not significantly differ, for the given grid selection.

7 Numerical illustration

In this section, we discuss some of our numerical experiments to illustrate the performance of the discretization procedures and of the adopted solvers. In the following we report the CPU time required to reach a desired accuracy, which is measured in terms of the relative residual

$$\text{Res} := \frac{\|R_k\|_F}{\|F\|_F} \tag{7.1}$$

where $\|\cdot\|_F$ is the Frobenius norm. For both theoretical purposes and approximation quality evaluation, the relative error norm is also of interest,

$$\text{Err} := \frac{\|U_k - U_\star\|_F}{\|U_\star\|_F} \tag{7.2}$$

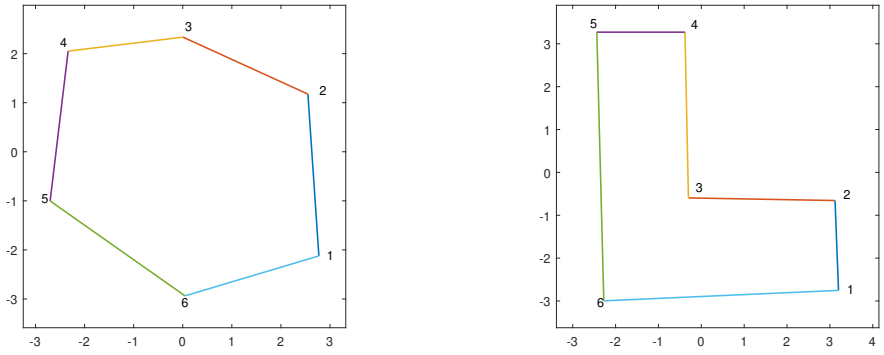

Fig. 3: The physical domain $\mathscr{P}$. Left: convex polygon. Right: L-shaped domain.

where $U_\star$ is the exact algebraic solution, obtained by solving the discretized problem by a direct solver, while U_k is the current approximation.

For simplicity of exposition, in all reported experiments we take $n_1 = n_2 = n$ and $f(x, y) \equiv 1$, and except where explicitly stated, we consider two different physical domains: a convex polygon and an L-shaped domain (cf. Fig. 3).

A sample of the commands used to create the mapping (via the SC Toolbox) and the associated cartesian grid is given by the following commands:

```
p=polyedit; % draw the polygon using the Toolbox Editor
t=[1 2 4 6]; % select the `control' vertices to build SC map
f=rectmap(p,t); % generate the SC map
a1=eval(inv(f),p.vertex(4));
a2=eval(inv(f),p.vertex(1)); % define the \xi-interval
b1= 0;
b2=imag(eval(inv(f),p.vertex(2))); % define the \eta-interval
plot(f) % plot the resulting grid on the polygon
n=2^6; % select the number of nodes on each edge
[xi,eta]=meshgrid(linspace(a1,a2,n),linspace(b1,b2,n)); % compute the nodes
df=evaldiff(f,xi+1i*eta); % compute the derivative of the SC map at the nodes
```

The computational cost, all concentrated in the function `rectmap`, significantly varies depending on the choice of the vertices in `t`. In some unfortunate cases, the map computation was too expensive to be completed, while for the majority of choices of `t` it took a small portion of the overall cost. We did not make any changes to the default parameters so as to ameliorate this occasional problem.

The reported numerical results were obtained with the choice of discretization grid that gave us consistently the best results. A detailed discussion on the grid selection is postponed to Section 8.

A few comments on the implementation details of the considered algorithms are in order.

The stationary iteration requires a Lyapunov solver at each iteration. Whenever the coefficient matrices are the same, these are eigendecomposed once for all, and the Lyapunov solution is explicitly written down (see, e.g., [35, Sect. 4]). Otherwise, the call to the Matlab function `lyap` is performed. The residual matrix is computed explicitly. All these computations are affordable for $n = \mathcal{O}(10^d)$ with d moderate, say $d \leqslant 3$, depending on the available machine. This is not restrictive for a two-dimensional problem, and a good accuracy of the sought after solution is obtained already for $n \approx 1000$.

Preconditioned restarted GMRES(m) is restarted every $m = 20$ iterations, as commonly done [32]. Different preconditioning strategies could be employed. For simplicity, here we consider two alternatives: (i) incomplete LU preconditioning[6], after permutation of the coefficient matrix rows and columns with symmet-

6 ILU factorization with threshold and pivoting was employed, with droptol tolerance 10^{-3}.

Tab. 1: Example of Section 7.1 in the L-shaped domain.

	direct	stationary iteration		ILUT GMRES(20)		Lyap GMRES(20)	
n	time	#iter	time	#rest	time	#rest	time
64	0.007	20	0.016	1(6)	0.106	1(4)	0.033
128	0.047	19	0.027	1(11)	0.107	1(5)	0.027
256	0.183	19	0.121	2(1)	0.552	1(5)	0.124
512	0.677	18	0.617	3(12)	4.741	1(5)	0.683
1024	4.348	18	4.481	6(19)	53.686	1(5)	4.666

ric minimum degree ordering; (ii) operator preconditioning, which consists of solving a Lyapunov equation with a portion of the coefficient matrix, typically the terms associated with the second order operator [29].

7.1 A reaction–diffusion problem

Consider the problem discussed in Section 5.1 with $\beta = 1$, that is

$$-\Delta u + u = f, \quad (x, y) \in \mathscr{P}$$

with f identically one. The discretized problem takes the form in (5.4), with $G_6 = J$ and $F = J \circ \mathbf{1} = J$, where we recall that the (i, j)th entry of J corresponds to $\mathscr{J}(\xi_i, \eta_j)$.

Table 1 shows the results of solving (5.4) in the L-shaped domain, for an SC grid taking as reference vertices the nodes [1 2 4 6] (ordered counterclock-wise), where the first node is $(3.2010, -2.7532)$. All iterations also checked the true error, and at completion the method had an error Err below 10^{-5}. In general, the relative residual norm in (7.1) was smaller or significantly smaller than the relative error norm in (7.2).

As expected, the direct method (Matlab backslash [27]) becomes more and more expensive as n grows, having to solve an $n^2 \times n^2$ system. Here and later on, we stress that comparing with the direct method is unfair, as the direct method employs a built-in Matlab function at its highest efficiency, whereas all other methods mainly rely on interpreted commands in Matlab. Comparisons between the stationary iteration and restarted preconditioned GMRES is more appropriate. The number of GMRES iterations in the last restart is reported in parentheses. For this simple operator, taking the second-order operator as preconditioner is clearly an advantage over a classical incomplete LU preconditioning, and this is also shown by the good performance of the stationary iteration, which uses the same operator. We also observe that the number of iterations for these two methods is independent of the number of grid nodes. This is related to the fact that the iteration operator (or preconditioned operator) is given (in Kronecker form) by

$$\mathcal{L}^{-1}(\mathcal{L} + \mathcal{G}) = I + \mathcal{L}^{-1}\mathcal{G}$$

where $\mathcal{L}$ accounts for the Laplace operator while the diagonal matrix $\mathcal{G}$ collects the Jacobian determinant entries. For a dominant Laplace operator, the eigenvalues of $I + \mathcal{L}^{-1}\mathcal{G}$ tend to tightly cluster, so that convergence of Krylov subspace methods can be faster [28, Ch. 4].

We ran the methods for the convex domain as well, with the direct and GMRES methods behaving the same as for the L-shaped domain. On the other hand, the stationary iteration failed to converge. This appears to be due to the different determinant, $\mathscr{J}$, whose discretized version J provides different spectral properties of the iteration matrix in this case.

Tab. 2: Example of Section 7.2 in the convex polygon, with nodes [1 2 3 5].

	direct	Projection method		ILUT GMRES(20)		Lyap GMRES(20)	
n	time	space dim	time	#rest	time	#rest	time
64	0.056	29	1.320	1(9)	0.073	3(16)	0.079
128	0.114	42	5.867	1(17)	0.166	4(17)	0.340
256	0.519	46	10.481	2(20)	1.066	6(12)	1.924
512	1.995	59	28.983	5(17)	13.927	8(18)	17.787
1024	11.738	66	82.368	14(2)	119.030	11(18)	135.400

Tab. 3: Example of Section 7.2 in L-shaped domain, with nodes [1 2 4 6].

	direct	Projection method		ILUT GMRES(20)		Lyap GMRES(20)	
n	time	space dim	time	#rest	time	#rest	time
64	0.019	13	0.241	1(9)	0.083	5(17)	0.146
128	0.097	23	0.698	1(16)	0.153	10(20)	1.291
256	0.467	33	2.668	2(18)	0.868	25(12)	7.634
512	1.995	50	13.700	4(13)	6.917	33(18)	77.751
1024	10.334	70	100.430	9(2)	72.063	— (—)	—

7.2 A simple elliptic problem

Consider the partial differential equation

$$\begin{cases} -(x^2+1)u_{xx} - (y^2+1)u_{yy} = 1, & (x,y) \in \mathscr{P} \\ u = 0, & (x,y) \in \partial\mathscr{P} \end{cases} \tag{7.3}$$

in the above two physical domains. The equation has non-constant coefficients, leading to the presence of the Jacobian determinant in the discretized matrix equation, as shown in (3.6). The numerical results for the convex polygonal domain are listed in Table 2, showing that for the finest grid the projection method performs the best in terms of CPU time. This is related to the fact that the GMRES-based methods are sensitive to the grid size, requiring a growing number of iterations as the grid is refined. On the other hand, the space dimension of the Galerkin method — and thus its computational cost — grows only mildly with increasing grid fineness. The results obtained with our experiments in the L-shaped domain are reported in Table 3. As before, timings for the direct method should not be compared with those of the other non-built-in algorithms. Note that GMRES preconditioned by the Lyapunov operator in this case is not effective, taking too long for the finest grid; it was thus stopped prematurely. This could be predicted as in this case the non-constant operator coefficients are not well approximated by the unit constant value. The projection method behaves quite well, using significantly lower memory than GMRES with a quite dense preconditioner.

7.3 A convection–diffusion problem

We consider the PDE

$$-u_{xx} - u_{yy} + \mathbf{w}^T \cdot \nabla u = f, \qquad (x,y) \in \mathscr{P} \tag{7.4}$$

with $\mathbf{w}^T = (\omega, 0)$ and homogeneous (zero) Dirichlet boundary conditions, in the same polygonal convex domain considered earlier, using $t = [1\ 2\ 3\ 5]$ as vertices. Discretization leads to the matrix equation in (5.13).

Here we analyze how the performance of the iterative algebraic solvers varies as ω increases, so as to make the problem more convection dominated. To this end, we focus on number of iterations, rather than on CPU times. The results are reported in Table 4. We clearly see that GMRES(20) with the operator preconditioner is insensitive to the meshsize variation, and it is only marginally affected by the increase of ω. ILUT

ω	n	Projection method space dim	ILUT GMRES(20) #rest(its)	Lyap GMRES(20) #rest(its)
	128	28	1(13)	1(9)
1	256	36	2(13)	1(9)
	512	42	5(6)	1(9)
	1024	45	9(16)	1(9)
	128	40	1(9)	2(19)
10	256	48	1(17)	2(19)
	512	54	7(19)	2(19)
	1024	67	14(4)	2(19)
	128	52	1(6)	6(8)
30	256	62	1(11)	6(7)
	512	80	2(5)	6(6)
	1024	87	9(14)	6(6)

Tab. 4: Performance of iterative solvers on the convection–diffusion problem in (7.4) for the convex domain.

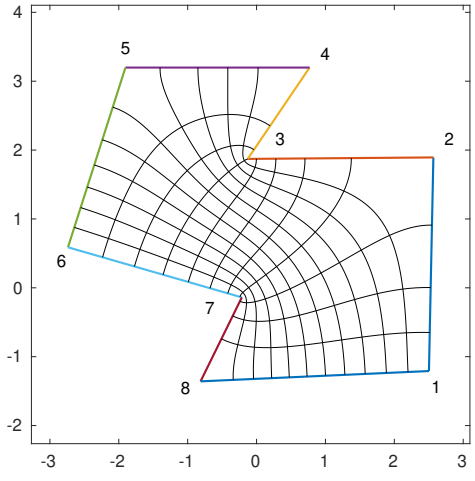

Fig. 4: General non-convex polygon. The physical domain $\mathscr{P}$ and an SC grid.

preconditioning does not share similar properties in terms of meshsize, while analogous results can be observed as ω varies. These results are similar to those observed when using operator preconditioners with other discretization approaches. It is interesting, however, that here the conformal mapping affects the convection terms in a different way than in other discretizations, as the partial derivatives X_ξ and X_η explicitly arise.

The projection method seems to be the most sensitive method to both meshsize and convection. On the other hand, it should be noticed that the projection method requires $O(n \cdot (\text{space dim}))$ memory allocations, which in the worst case amounts to $O(87 \cdot 1024)$, whereas GMRES(20) requires $O(20 \cdot n^2)$ allocations, which goes up to $O(20 \cdot 1024^2)$ allocations for the largest problem considered, without taking into account the memory required for the factors in the ILUT preconditioning. Hence, if memory is a constraint, GMRES methods on the vector problem can be largely penalized.

To complete this experimental analysis, we consider a different non-convex domain, displayed in Fig. 4, for the vertex choice $t = [1\ 4\ 6\ 8]$ (the first vertex is the most south-east one). Schwarz–Christoffel mappings are particularly appropriate to handle this type of complex domain. Table 5 reports the results of our experiments for problem (7.4) and $\omega = 1$, the number of iterations, space dimensions and CPU times to be compared with those in Table 4 for the same choice of ω. The performance of all methods is completely analogous to that for the polygonal convex domain, emphasizing that once a good vertex choice is made, the performance of this discretization procedure is algebraically robust. Indeed, for the projection method we found the selection of the 'control' vertices to be quite crucial, and this is further discussed in the next section.

Tab. 5: Performance of iterative solvers on the convection–diffusion problem in (7.4) for $\omega = 1$, with the non-convex domain in Fig. 4.

	Projection method		ILUT GMRES(20)		Lyap GMRES(20)	
n	space dim	time	#rest(its)	time	#rest(its)	time
128	34	2.4	1(14)	0.1	1(8)	0.06
256	44	7.1	2(13)	0.7	1(8)	0.2
512	50	16.3	4(16)	7.4	1(8)	0.6
1024	57	51.7	10(19)	87.0	1(8)	9.1

8 Analysis of the grid selection

In our numerical experiments with the Galerkin method we found that the method was overly sensitive to the grid choice, or more precisely, to the selection of the four control vertices to be mapped to the four rectangle vertices. In particular, the number of iterations, and thus the space dimension of V_k and W_k, varied quite significantly for different vertex selections. We quote a paragraph from Trefethen and Driscoll's book [15] as a warning for such an occurrence:

> An inspection of many of the figures in this book makes it clear that the size of equally spaced cells in the computational domain can vary greatly and rapidly in the physical domain [...]. This variation makes a certain amount of physical sense, at least in some contexts, but is undesirable in many applications. A particular problem [...] in SC mapping is the presence of singularities in f due to the corners. As a mapping, the SC formula handles corners elegantly and completely, but here they remain a significant challenge in the computational domain.

Before we continue we wish to stress that we did not encounter any performance sensitivity in using the vector oriented GMRES. Hence, we conjecture that the computational sensitivity arises when the *matrix* properties of $\mathscr{J}$ play a role, whereas the vectorized version of $\mathscr{J}$ only influences the computation with its positive entries.

We thus proceed with a deeper analysis of the role of $\mathscr{J}$ in the algorithmic performance of the Galerkin approach for the convex polygon. A corresponding analysis was performed for the other considered domains with similar results, hence these are not reported. We focus our analysis on the Poisson equation in (4.2), where $\mathscr{J}$ only enters in the right-hand side and it is easier to analyze. Moreover, we distinguish between a good grid, for which we experienced fast convergence, and a bad grid, which led to a significant delay as the discretization is refined.

With the good grid $t = [1\ 2\ 3\ 5]$, for instance, the Galerkin method applied with $n = 2^{10}$ selects an approximation to the right-hand side of rank 17, and with a space of dimension 45 reaches the accuracy required by (7.1). On the contrary, with the bad grid $t = [1\ 2\ 3\ 6]$, in spite of a smaller rank for the right-hand side approximation (rank 11), the method does not reach completion within a space of dimension 100. We stress that the singular values of $\mathscr{J}$ in the two cases show similar patterns and they did not appear to play any role in the different behavior.

Figure 5 displays the residual norm history as the space dimension grows, for the two cases, illustrating the significantly different slopes in the convergence curve. The plot also includes the error F-norm history (see (7.2)) for the two cases. We recall that the Galerkin method minimizes the $\mathcal{S}$-norm of the error for the given approximation space (see Proposition 6.4). Thus, the error behavior displayed in the plot, though in a different norm, indicates that the space generated by starting with vectors from the bad grid does not provide as good information as for the good grid. In passing, we also observe that the error is, in both cases, much smaller than the residual norm would predict. This appears to be unrelated to the low rank truncation of the right-hand side used to build the approximation space, but rather to classical results on the discrepancy between error and residual norms. For this specific problem, this is given by

$$\|R_k\|_F \leqslant \|I \otimes T_1 + T_2 \otimes I\| \, \|U_k - U_\star\|_F.$$

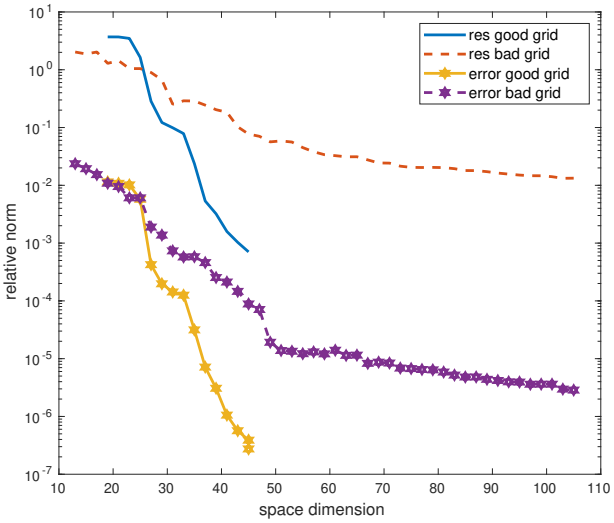

Fig. 5: History of relative residual and error norm convergence for the good grid $t = [1\,2\,3\,5]$ and for the bad grid $t = [1\,2\,3\,6]$.

Tab. 6: Some sample grids discretizing the polygon, using $n = 2^{10}$ grid nodes on each side of the rectangle in (ξ, η).

good quality	vertices	aspect ratio	max df	maxmin ratio
✓	[1 2 3 5]	1.10	16.33	389.4
✓	[1 3 5 6]	1.18	14.71	258.8
✓	[1 2 4 6]	1.06	12.30	106.7
✗	[1 2 3 6]	1.17	19.60	413.9
✗	[3 4 5 6]	1.14	25.21	769.1
✗	[1 2 3 4]	1.31	15.44	678.0

Since $T_1 = T_2$ and they are tridiagonal, symmetric and Toeplitz matrices of size n as defined in Section 3.1, we get

$$\|I \otimes T_1 + T_2 \otimes I\| = \lambda_{\max}(I \otimes T_1 + T_2 \otimes I) = 2\lambda_{\max}(T_1) = \frac{2}{h_1^2}\left(2 + 2\cos\left(\frac{\pi}{(n-1)}\right)\right).$$

For $n = 2^{10}$, it thus holds that $\|I \otimes T_1 + T_2 \otimes I\| \approx 8 \cdot 10^6$, which largely accommodates the gap between error and residual norms observed in the plot.

8.1 Action of good and bad grids

Table 6 reports some grid information for different selections of the polygon vertices that are used by the mapping $\mathfrak{h}$ to construct the four rectangle pre-vertices. More precisely, for each vertex selection, the aspect ratio is the ratio between the largest and smallest rectangle edges; `max df` is the maximum absolute value of the Jacobian determinant square root, $\max_{\xi,\eta} \sqrt{x_\xi^2 + x_\eta^2}$ for the adopted grid; `maxmin ratio` is the ratio between the longest and shortest grid edges on the polygon boundary for the chosen grid. The first three grid selections in Table 6 lead to good performance of the Galerkin method, the second three choices lead to a delayed convergence. The only parameter that seems to be in favor of the good grids is the maxmin ratio, which tends to be smaller than for the bad grids. Figure 6 shows the grid mapped by the discretization of the corresponding rectangle via the SC Toolbox, for two choices of vertices (a good grid on the left, a bad grid on the right).

The Schwarz–Christoffel mapping associated with these two grids leads to the Jacobian determinant $\mathcal{J} = \mathcal{J}(\xi_i, \eta_j)$, $i, j = 1, \ldots, n$, in Fig. 7. The peaks refer to singularity approximations corresponding to the disregarded polygon vertices. For the good choice $t = [1\,2\,3\,5]$ the missing vertices [4 6] are not consecutive, therefore each corresponds to a peak on a different axis direction. Two close peaks on a single edge occur in the bad choice $t = [1\,2\,3\,6]$, corresponding to the disregarded vertices [4 5]. We believe this is a key fact:

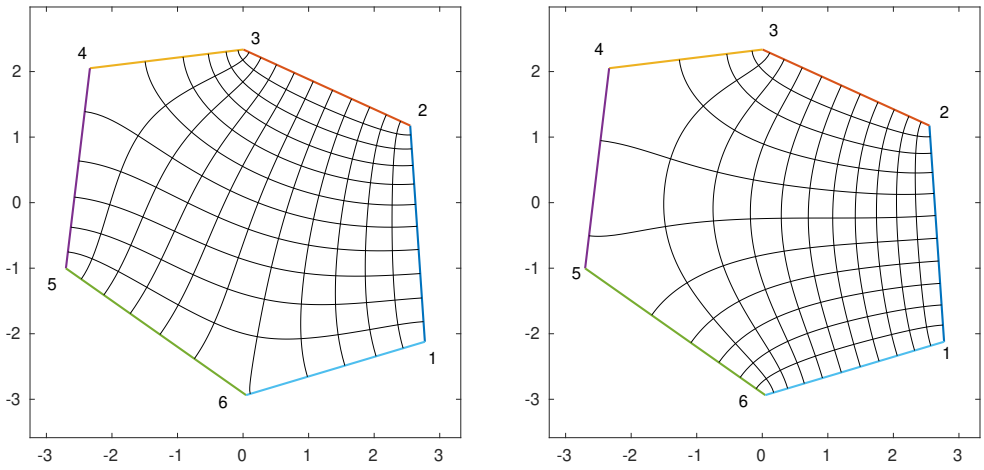

Fig. 6: Typical meshes for the polygonal domain (coarse grids are shown). Left: Good grid $t = [1\ 2\ 3\ 5]$. Right: Bad grid $t = [1\ 2\ 3\ 6]$.

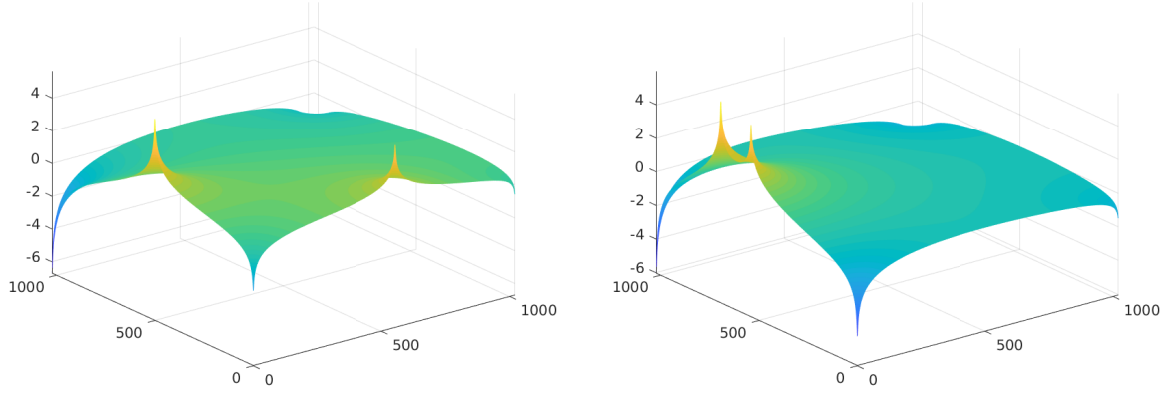

Fig. 7: Entry magnitudes of $\mathscr{J}$ for the (ξ, η) domain (log scale). Left: Good grid $t = [1\ 2\ 3\ 5]$. Right: Bad grid $t = [1\ 2\ 3\ 6]$.

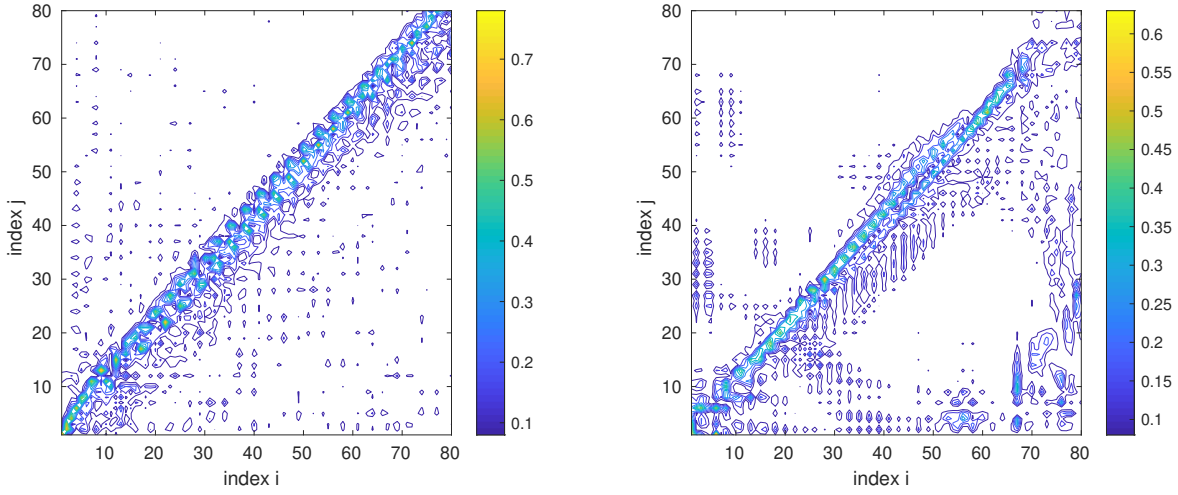

Fig. 8: Contour plot of the entries $|\Theta_{i,j}|$ of the 'angle matrix' $\Theta = P^T Z$, where P, Z are the first 80 left and right singular vectors of J, respectively. Left: Good grid $t = [1\ 2\ 3\ 5]$. Right: Bad grid $t = [1\ 2\ 3\ 6]$.

alternating disregarded vertices lead to a better approximation of the solution for this particular problem. We argue that this different behavior may affect the choice of the starting blocks generating the right and left approximation spaces, $\mathcal{V}_k$, $\mathcal{W}_k$, which in turn may lead to the generation of poor spaces.

We start by recalling that if $T_1 = Q\Lambda Q^T$ is the eigenvalue decomposition of T_1 with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, then

$$U = Q\left((Q^T J Q) \oslash (\Lambda \otimes I + I \otimes \Lambda)\right) Q^T \tag{8.1}$$

where '$\oslash$' denotes elementwise division. For this problem the solution nonsymmetry is only due to J. If J were symmetric, then in the approximation $V_k Y_k W_k^T \approx U$, the spaces $\mathcal{V}_k$, $\mathcal{W}_k$ could be taken to be equal. Moreover, due to the expression in (8.1), a good approximation to U can be obtained only if the eigencomponents corresponding to the smallest eigenvalues of T_1 are well approximated in both spaces $\mathcal{V}_k$, $\mathcal{W}_k$. Since these eigencomponents are discretizations of smooth functions, it would be desirable that this type of vectors could be readily detected in the spaces, hence the two spaces should behave somewhat similarly. Let us see how these properties transfer into spectral requirements on J. Let $J = P\Sigma Z^T$ be the singular value decomposition of J. Some sort of symmetry in J is reflected in the left and right singular vectors of J behaving similarly. This implies that taking the first left (resp., right) singular vectors to build $\mathcal{V}_k$ (resp., $\mathcal{W}_k$) then the generated spaces would be similar. This behavior is displayed in Fig. 8 for J stemming from the good grid (left) and from the bad grid (right). If J were symmetric, then $P^T Z = I$. This is not the case for either grid, however the plots show that for the good grid the neighboring vectors in P are close to the corresponding vectors in Z; this is not the case for the bad grid, as vectors in P have large components onto many column vectors in Z. By taking the first columns of P and Z to build the two approximation spaces, the resulting spaces will significantly differ for the bad grid.

The presence of peaks in J results into a corresponding structure in the vectors of P and Z. This is expected since the first ℓ singular triplets of J provide the best approximation to J of rank ℓ (see [21, Th. 2.4.8]). Hence, if the first columns of P and Z have a non-smooth behavior (to match the occurrence of several peaks), these will not be able to quickly approximate the smooth eigencomponents of the operator T_1, leading to slow convergence. In other words, the smoothness of the operator T_1 should be well captured by the starting vectors generating $\mathcal{V}_k$, $\mathcal{W}_k$, for a fast approximation to take place. The bad grid (right plot of Fig. 7) has several peaks on one side of the grid, and these are only taken care of by the left singular vectors. This seems to imply that $\mathcal{V}_k$ and $\mathcal{W}_k$ will be very different, and that the space $\mathcal{W}_k$ will only slowly approximate the leading eigencomponents in the solution U, causing a delay.

We also experimented with polygons having a larger number of vertices. It was observed that better performance was obtained when unselected vertices were mapped close to each other on the rectangle edges — which clustered the peaks in the Jacobian determinant — a somewhat positive effect of the SC mapping crowding phenomenon. This reflected in smoother leading singular vectors, strengthening the previous argument.

In summary, our very preliminary analysis seems to suggest that, to avoid a very oscillatory behavior in the starting blocks of the approximation spaces, the control vertices should distribute somewhat evenly among all polygon vertices, so as to avoid missing too many consecutive vertices. Moreover, clustering of the unselected vertices on the rectangle edges appears to be beneficial. The problem clearly deserves further deepening as a building block of the discretization procedure for the matrix-oriented formulation, given its role in the performance of projection methods.

9 Conclusions

We have analyzed the matrix computational problems associated with the discretization of linear elliptic partial differential equations defined in polygonal domains by means of the Schwarz–Christoffel mappings. The matrix oriented formulation leads to special linear matrix equations, involving Hadamard products, which make the use of matrix equation solvers quite challenging. On the other hand, iterative vector based methods such as GMRES preconditioned by the higher-order operator seem to behave quite well, with a number of iterations that seems to be mesh-independent, and quite insensitive to possible corner singularities of the

mapping. If memory allocations are an issue, then matrix-oriented methods should be preferred: a working strategy with several implementation details is provided.

Our use of the Schwarz–Christoffel mappings was surely encouraged by the presence of an easy to use Matlab Toolbox. However, it is well known that these mappings are restricted to two-dimensional domains. Hence, the generalization of our methodology to three-dimensions would certainly need to significantly extend this setting, or employ different transformations for the generation of structured grids.

Acknowledgment: We thank two anonymous reviewers for their careful reading and for several helpful remarks. The second author is a member of Indam-GNCS. Its support is gratefully acknowledged. Part of this work was also supported by the Grant AlmaIdea 2017-2020 – Università di Bologna. The first author is funded by the China Scholarship Council (Contract No. 201906180033) and by the National Natural Science Foundation of China (Grant Nos. 11471150, 11401281). Her work was performed during her visit at the Università di Bologna, Italy.

References

[1] R. H. Bartels and G. W. Stewart, Solution of the matrix equation $AX + XB = C$, *Commun. ACM*, **15** (1972), 820–826.
[2] M. Z. Bazant, Conformal mapping of some non-harmonic functions in transport theory, *Proc. R. Soc. Lond. A*, **460** (2004), 1433–1452.
[3] P. Benner and T. Damm, Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems, *SIAM J. Control Optim.*, **49** (2011), 686–711.
[4] P. Benner, J.-R. Li, and T. Penzl, Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems, *Numer. Linear Alg. Appl.*, **15** (2008), 1–23.
[5] M. Benzi, Preconditioning techniques for large linear systems: a survey, *J. Comput. Phys.*, **182** (2002), 418–477.
[6] C. Canuto, V. Simoncini, and M. Verani, Contraction and optimality properties of an adaptive Legendre–Galerkin method: the multi-dimensional case, *J. Sci. Comput.*, **63** (2014), 769–798.
[7] J. E. Castillo, ed., *Mathematical Aspects of Numerical Grid Generation*, Frontiers in Applied Mathematics, SIAM, 1991.
[8] S. Chakravarthy and D. Anderson, Numerical conformal mapping, *Math. Comput.*, **33** (1979), 953–969.
[9] N. V. Challis and D. M. Burley, A numerical method for conformal mapping, *IMA J. Numer. Analysis*, **2** (1982), 169–181.
[10] T. Damm, Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations, *Num. Linear Alg. Appl.*, **15** (2008), 853–871.
[11] T. Damm, P. Benner, and J. Hauth, Computing the stochastic H^∞-norm by a Newton iteration, *IEEE Control Systems Letters*, **1** (2017), No. 1, 92–97.
[12] T. Damm, K. Sato, and A. Vierling, Numerical solution of Lyapunov equations related to Markov jump linear systems, *Numer Linear Alg. Appl.*, **25** (2017), No. 6, e2113.
[13] T. A. Driscoll, Algorithm 756: A MATLAB toolbox for Schwarz–Christoffel mapping, *ACM Trans. Math. Softw.*, **22** (1996), 168–186.
[14] T. A. Driscoll, Algorithm 843: Improvements to the Schwarz–Christoffel toolbox for MATLAB, *ACM Trans. Math. Softw.*, **31** (2005), 239–251.
[15] T. A. Driscoll and L. N. Trefethen, *Schwarz–Christoffel Mapping*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2002.
[16] V. Druskin and V. Simoncini, Adaptive rational Krylov subspaces for large-scale dynamical systems, *Systems Control Letters*, **60** (2011), 546–560.
[17] N. S. Ellner and E. L. Wachspress, New ADI model problem applications, In: *Proc. of 1986 ACM Fall Joint Computer Conference*, Dallas, Texas, United States, IEEE Computer Society Press, Los Alamitos, CA, 1986, pp. 528–534.
[18] H. C. Elman, D. J. Silvester, and A. J. Wathen, Finite Elements and Fast Iterative Solvers, with Applications in Incompressible Fluid Dynamics, 2nd ed., Oxford University Press, Oxford, 2014.
[19] M. Farrashkhalvat and J. P. Miles, *Basic Structured Grid Generation*, Butterworth & Heinemann, 2003.
[20] B. Fornberg, A numerical method for conformal mappings, *SIAM J. Sci. Stat. Comp.*, **1** (1980), 386–400.
[21] G. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 4th ed., 2013.
[22] L. H. Howell and L. T. Trefethen, A modified Schwarz–Christoffel transformation for elongated regions, *SIAM J. Sci. Stat. Comput.*, **11**, (1990), No. 5, 928–949.
[23] D. C. Ives, Conformal grid generation, *Appl. Math. Comput.*, **10-11** (1982), 107–135.
[24] E. Jarlebring, G. Mele, D. Palitta, and E. Ringh, Krylov methods for low-rank commuting generalized Sylvester equations, *Numer. Linear Alg. Appl.*, **25** (2018), No. 6, e2176.

[25] P. M. Knupp and S. Steinberg, *The Fundamentals of Grid Generation*, Knupp, 1992.
[26] C. W. Mastin and J. F. Thompson, Quasiconformal mappings and grid generation, *SIAM J. Sci. Stat. Comp.*, **5** (1984), 305–310.
[27] The MathWorks, Inc., *MATLAB 7*, r2017b ed., 2017.
[28] M. A. Olshanskii and E. E. Tyrtyshnikov, *Iterative Methods for Linear Systems, Theory and Applications*, SIAM, 2014.
[29] D. Palitta and V. Simoncini, Matrix-equation-based strategies for convection–diffusion equations, *BIT Numer. Math.*, **56** (2016), 751–776.
[30] D. Palitta and V. Simoncini, Optimality properties of Galerkin and Petrov–Galerkin methods for linear matrix equations, *Vietnam J. Math.*, **48** (2020), 791–807.
[31] C. E. Powell, D. Silvester, and V. Simoncini, An efficient reduced basis solver for stochastic galerkin matrix equations, *SIAM J. Sci. Comp.*, **39** (2017), A141–A163.
[32] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, 2nd ed., 2003.
[33] G. Sangalli and M. Tani, Isogeometric preconditioners based on fast solvers for the Sylvester equation, *SIAM J. Sci. Comput.*, **38** (2016), A3644–A3671.
[34] S. D. Shank, V. Simoncini, and D. B. Szyld, Efficient low-rank solutions of generalized Lyapunov equations, *Numerische Mathematik*, **134** (2016), 327–342.
[35] V. Simoncini, Computational methods for linear matrix equations, *SIAM Review*, **58** (2016), 377–441.
[36] D. Sorensen and Y. Zhou, Direct methods for matrix Sylvester and Lyapunov equations, *J. Appl. Math.*, **2003** (2003), No. 6, 277–303.
[37] G. T. Symm, An integral equation method in conformal mapping, *Numer. Math.*, **9** (1966), 250–258.
[38] J. F. Thompson, *Numerical Grid Generation*, Elsevier, Amsterdam, 1982.
[39] J. F. Thompson, Z. U. Warsi, and C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*, Elsevier, North-Holland, Inc., USA, 1985.
[40] L. N. Trefethen, Numerical computation of the Schwarz–Christoffel transformation, *SIAM J. Sci. Comput.*, **1** (1980), 82–102.
[41] L. N. Trefethen, Numerical conformal mapping with rational functions, *Comput. Methods Function Theory* **20** (2020), No. 3, 369–387.
[42] E. L. Wachspress, *Iterative Solution of Elliptic Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1966.