An AI-aided algorithm for multivariate polynomial reconstruction on Cartesian grids and the PLG finite difference method

Qinghai Zhang*[1,2], Yuke Zhu[1], and Zhixuan Li[1]

[1]School of Mathematical Sciences, Zhejiang University, 866 YuHangTang Road, Hangzhou, Zhejiang Province, 310058, China

[2]Shanghai Institute for Advanced Study of Zhejiang University, Shanghai AI Laboratory, Shanghai, 200000, China

Abstract

Polynomial reconstruction on Cartesian grids is a key problem in developing finite difference methods as well as in many other engineering applications, yet it is still an open problem how to construct for a finite subset K of $\mathbb{Z}^{\mathsf{D}}$ a lattice $\mathcal{T} \subset K$ so that multivariate polynomial interpolation on this lattice is unisolvent. In this work, we solve this open problem of poised lattice generation (PLG) via an interdisciplinary research of approximation theory, abstract algebra, and artificial intelligence (AI). Specifically, we focus on the triangular lattices in approximation theory, study group actions of permutations upon triangular lattices, prove an isomorphism between the group of permutations and that of triangular lattices, and dynamically organize the AI state space of permutations so that a depth-first search of poised lattices has optimal efficiency. Based on this algorithm, we further develop the PLG finite difference method that retains the simplicity of Cartesian grids yet overcomes the disadvantage of legacy finite difference methods in handling irregular geometries. Results of various numerical tests demonstrate the effectiveness of our algorithm and the simplicity, efficiency, and fourth-order accuracy of the PLG finite difference method.

Keywords: multivariate polynomial interpolation, finite difference methods, data reconstruction, artificial intelligence, depth-first search with backtracking, poised lattice generation

*Corresponding author: qinghai@zju.edu.cn

1 Introduction

In numerically solving partial differential equations (PDEs), the simplest and historically oldest method is probably the finite difference (FD) method, which mainly consists of replacing spatial derivatives in the PDE with FD formulas and solving the resulting system of ordinary differential equations (ODEs) by a time integrator. Supported by an extensive body of theory, FD methods are robust, efficient, and accurate for a variety of PDEs [31, 16, 19].

Legacy FD methods are often criticized as ill-suited for complex geometries. Indeed, FD methods rely heavily on the geometric regularity of the underlying Cartesian grids, and multidimensional FD formulas are usually based on tensor products of one-dimensional formulas. These regularities of FD stencils severely limit the application of FD methods to irregular domains.

1.1 Motivations from accurately and efficiently solving PDEs on irregular domains with Cartesian grids

There are two approaches that could alleviate the unfitness of legacy FD methods for irregular domains while retaining the simplicity of Cartesian grids.

In the first approach, classical FD stencils are retained for *point* values at grid points in the interior of the domain while special treatments are adopted at grid points near the irregular boundary. Successful examples of this approach include the immersed boundary method [25, 22], the immersed interface method [17, 18], the ghost cell approach [21, 33, 20, 40, 37], and many others such as the MIB method [41] and the KFBI method [36].

The second approach concerns a finite volume (FV) formulation, where the unknowns are *averaged* values over the rectangular cells of the Cartesian grid. Similar to the first approach, spatial discretization away from the irregular boundary is based on symmetric stencils while cells near the boundary are treated differently. In both approaches, this strategy of separately treating 'regular' grids and 'irregular' grids can be made very efficient by exploiting the fact that the irregular grids requiring more involved treatments form a set of codimension one.

A popular example of the second approach is the cut-cell method, also known as the embedded boundary (EB) method, in which cells close to the boundary are cut by the irregular interface and averaged values are defined on the open region, i.e., the intersection of the cell and the problem domain; see, e.g., [14, 34, 11, 6, 24] and references therein. Previous EB methods are second-order accurate and it is only recently that fourth-order EB methods emerged [6, 24]. In these methods, FV formulas are derived for regular cells based on Taylor expansions while multivariate polynomials are fitted to

interpolate cell-averaged values near the irregular boundary in a least-square sense. Then spatial operators are discretized via integrating the derivatives of the fitted polynomials over irregular cells.

This work is motivated by the following observations and questions.

(Q.1) Most FD-based methods are only first- or second-order accurate. Can we develop, for irregular domains with arbitrarily complex topology and geometry, a new FD method that can be fourth- or higher-order accurate?

(Q.2) Treatments of the irregular boundary in most FD-based methods revolve around modifying one-dimensional FD formulas. As such, it is not clear how to generalize them to PDEs with cross derivatives such as $\frac{\partial^2 u}{\partial x \partial y}$. Can we design a new FD method whose formulation is completely decoupled from the specific form of the PDE?

(Q.3) For fourth-order EB methods, it has never been rigorously proven that the linear system resulting from polynomial reconstruction indeed admit a unique solution for coefficients of the polynomial. In fact, even a precise description of which cells get into the stencil of least squares is rare. Of course, one can keep adding nearby cells until the linear system becomes solvable. But blindly expanding the least-square stencil may lead to a large number of redundant points, deteriorating computational efficiency. As will be shown in Figure 7, it may also increase the condition number of the linear system by a large factor. Therefore, for the multivariate polynomial fitting over least square stencils, can we prove the unique solvability of the linear system for the polynomial coefficients? Can we give an explicit and systematic construction of polynomial-fitting stencils so that the number of cells in any stencil is the minimum? Furthermore, can we exert fine control over the least squares? or, more precisely, can we guarantee the least condition number and no redundant cells?

(Q.4) The algorithmic complexity of EB methods and many FD-based methods increases rapidly as the dimensionality goes from two to a higher one. Can we suppress the algorithmic complexity of the new FD method so that the special treatment remains the same in two and higher dimensions?

In this paper, we give positive answers to all the above questions, via an intensive study on data reconstruction by multivariate polynomial interpolation.

Definition 1.1 (Lagrange interpolation problem (LIP) [1])**.** *Let* Π^D *denote the linear space of all* D*-variate polynomials with real coefficients. Given a subspace* V *of* Π^D*, a finite number of points* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in \mathbb{R}^D$ *and the same*

number of data $f_1, f_2, \ldots, f_N \in \mathbb{R}$*, the* Lagrange interpolation problem *seeks a polynomial* $f \in V$ *such that*

$$\forall j = 1, 2, \ldots, N, \ f(\mathbf{x}_j) = f_j, \tag{1}$$

where V *is called the* interpolation space *and* $\mathbf{x}_j$*'s the* interpolation sites.

A LIP with $\dim V = N$ is said to be *unisolvent* if for *any* given data $(f_j)_{j=1}^N$ there exists $f \in V$ satisfying (1); in this case we also say that the sites $(\mathbf{x}_j)_{j=1}^N$ are *poised* in V or they are poised with respect to a basis of V.

All our answers to (Q.1–4) are based on a novel and efficient solution to the open problem as follows.

1.2 The open problem of poised lattice generation (PLG) in $\mathbb{Z}^{\mathsf{D}}$

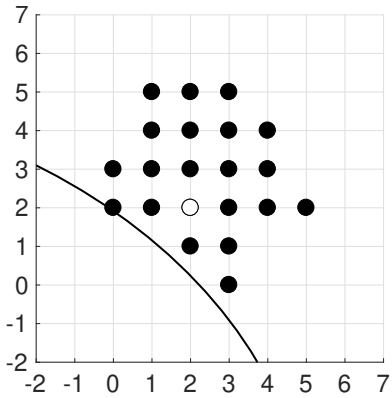
Definition 1.2 (PLG in $\mathbb{Z}^{\mathsf{D}}$)**.** *Given a finite set of feasible nodes* $K \subset \mathbb{Z}^D$*, a starting point* $\mathbf{q} \in K$*, and a degree* $n \in \mathbb{Z}^+$*, the* problem of poised lattice generation *is to choose a lattice* $\mathcal{T} \subset K$ *such that* $\mathcal{T}$ *is poised in* Π_n^D *and* $\#\mathcal{T} = \dim \Pi_n^D = \binom{D+n}{n}$*, where* Π_n^D *is the space of* D*-variate polynomials of total degree at most* n*.*

In this work, we limit values of n in Definition 1.2 to $n \leq 6$; see Section 1.3 for reasons of this limitation. Two examples of PLG are shown in Figure 1. The starting point $\mathbf{q}$ corresponds to the grid point at which the spatial operators are discretized, and the feasible set K can be customized according to the physics of the PDE. As illustrated in Figure 4, for the Laplacian operator one might want to choose K so that $\mathbf{q}$ is centered in K to reflect the isotropic nature of diffusion. In contrast, for the advection operator it is more appropriate to choose K to favor the upwind direction so that the domain of dependence of the advection is covered. Note that the problem in Definition 1.2 might not have a solution; in this case one can either shift/expand the feasible set K or decrease the degree n until the problem is solvable.

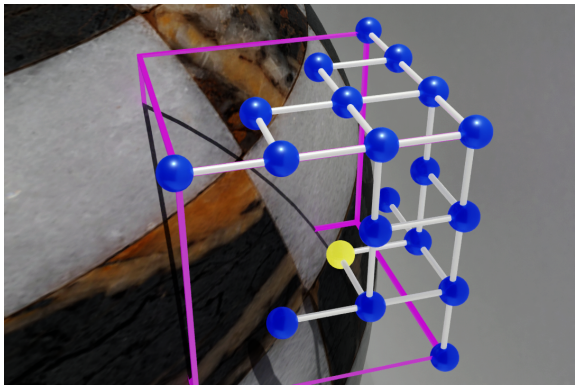
Suppose that a LIP is unisolvent with its interpolation space V spanned by basis functions $\phi_1, \phi_2, \ldots, \phi_N$. Then the *sample matrix*

$$M = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_N(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_N) & \phi_2(\mathbf{x}_N) & \cdots & \phi_N(\mathbf{x}_N) \end{bmatrix} \tag{2}$$

must be non-singular. Indeed, $\det M = 0$ would imply that the dimension of the range of M be less than N and that there exist some data $(f_j)_{j=1}^N$ for which the LIP has no solution. Therefore, the nonsingularity of the sample



(a) A poised lattice in Π_5^2 at the hollow dot $\mathbf{q}$.



(b) A poised lattice in Π_3^3 at the yellow ball $\mathbf{q}$.

Figure 1: Two solutions of the PLG problem, where the feasible set K is obvious.

matrix, the existence of a unique solution of the LIP, and the poisedness of the interpolation sites are equivalent conditions.

In the familiar case of $\mathsf{D} = 1$, the LIP is unisolvent if and only if its sites are pairwise distinct. In other words, any N pairwise-distinct points in $\mathbb{R}$ are poised in $V = \Pi_{N-1}^1$. In contrast, for $\mathsf{D} > 1$, it is much more difficult to decide whether a set of sites is poised in Π_n^{D}. For example, suppose six data are given at the sites

$$(5,0),\ (-5,0),\ (0,5),\ (0,-5),\ (4,3),\ (-3,4).$$

Then for the space $V = \Pi_2^2 = \mathrm{span}(1, x, y, x^2, y^2, xy)$, the sample matrix

$$M = \begin{bmatrix} 1 & 5 & 0 & 25 & 0 & 0 \\ 1 & -5 & 0 & 25 & 0 & 0 \\ 1 & 0 & 5 & 0 & 25 & 0 \\ 1 & 0 & -5 & 0 & 25 & 0 \\ 1 & 4 & 3 & 16 & 9 & 12 \\ 1 & -3 & 4 & 9 & 16 & -12 \end{bmatrix}$$

is singular since its first, fourth, and fifth columns are linearly dependent; indeed, all six sites are on the circle $x^2 + y^2 - 25 = 0$. As a crucial difference between univariate and multivariate polynomials, the latter usually vanishes on an *infinite* number of points. In general, a set of sites is poised if and only if the interpolation space does not contain a polynomial that vanishes on all the sites. This underlines a core difficulty of multivariate interpolation: *the poisedness of interpolation sites depends on their geometric configuration.*

In approximation theory, much effort has been directed to constructing poised lattices. In the classical paper by Chung and Yao [4], a set X of $N :=$

$\binom{\mathsf{D}+n}{n}$ sites in $\mathbb{R}^{\mathsf{D}}$ is said to satisfy the *condition of geometric characterization* in Π_n^{D} if for each site $\mathbf{x}_i$ there exist n distinct hyperplanes $G_{i,1}$, $G_{i,2}$, ..., $G_{i,n}$ such that

$$\forall i, j = 1, 2, \ldots, N, \qquad i \neq j \;\Leftrightarrow\; \mathbf{x}_j \in \cup_{\ell=1}^{n} G_{i,\ell}, \tag{3}$$

i.e., each $\mathbf{x}_i$ does not lie on any of these hyperplanes and all the other nodes in X lie on at least one of these hyperplanes. By constructing Lagrange fundamental polynomials that are 1 at one site and 0 at all other sites, they showed that the condition of geometric characterization guarantees a unique interpolating polynomial of degree at most n. Chung and Yao [4] also proposed two special sets of poised sites, called natural lattices and principal lattices (in this work a lattice refers to a set of interpolation sites whose cardinality equals the dimension of the space of interpolating polynomials), both of which satisfy the condition of geometric characterization. Following this idea of choosing intersections of appropriate hyperplanes, researchers have proposed other types of poised lattices such as $(\mathsf{D}+1)$-pencil lattices [15, 13], (fully) generalized principal lattices [1, 3, 5], Aitken-Neville sets [30, 2, 5], and lower sets [35, 28, 7]. See [9] for a review.

Unfortunately, the freedom of choosing *any* points in $\mathbb{R}^{\mathsf{D}}$ is always assumed in the aforementioned methods; but this freedom assumption in $\mathbb{R}^{\mathsf{D}}$ does not hold for the PLG problem in Definition 1.2.

To our best knowledge, there are no systematic solutions to Definition 1.2, and thus PLG is still an open problem.

1.3 The nature of the PLG problem

It is now a good time to discuss several key issues on the formulation of PLG in $\mathbb{Z}^{\mathsf{D}}$ so as to reveal its nature and to dispel potential doubts on Definition 1.2.

First, we explain why values of n are limited to $n \leq 6$ in Definition 1.2.

By the Runge phenomenon, polynomial interpolation on uniform grids diverges as the polynomial degree $n \to \infty$. Indeed, the condition number of the Vandermonde matrix (2) on uniform grids grows exponentially [10, p. 24]:

$$\mathrm{cond}_\infty M \approx \frac{1}{\pi} \exp\left[\frac{n}{4}(\pi + 2\ln 2) - \frac{\pi}{4}\right]. \tag{4}$$

Define the *Lebesgue constant* of the linear functional of interpolation as $\Lambda := \sup_{f \in \mathcal{C}([-1,1])} \frac{\|p\|_\infty}{\|f\|_\infty}$ where $f \in \mathcal{C}([-1,1])$ is the continuous function to be interpolated, p a polynomial interpolant, and p^* the best approximation of f. Then it can be shown [32, Thm 15.1] that a polynomial interpolant is good if and only if its Lebesgue constant is small, i.e.,

$$\|p - f\|_\infty \leq (1 + \Lambda)\|p^* - f\|_\infty. \tag{5}$$

Since $\Lambda(n) \geq \frac{2}{\pi}\log(n+1) + 0.5$ [32, Thm 15.2], it is not a good idea to seek the solution of PLG in $\mathbb{Z}^{\mathsf{D}}$ for a large n.

But how large is large? For the Vandermonde matrix, we have from (4) that $\mathrm{cond}_\infty M \approx 160, 8700, 1.0 \times 10^9$ for $n = 6, 10, 20$, respectively. As for the Lebesgue constant, we have $\Lambda \approx 7, 51$, and 4.2×10^{12} for n=7, 11, and 50, respectively [32, Chap. 15]. Of course these results are for one dimension, but they serve as a strong heuristic that multivariate polynomial interpolation on uniforms grids is fine if we impose a small upper bound on the total degree, say, $n \leq 6$. This is confirmed by results of numerical experiments in Section 6.

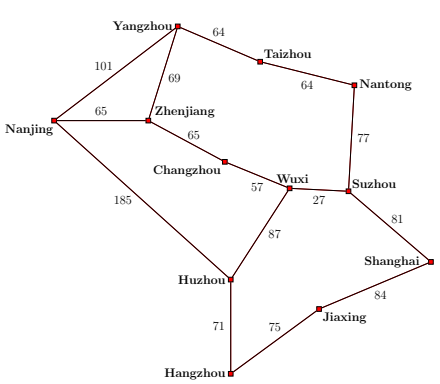
Second, we emphasize that the PLG problem, even with limited small values of n, is not trivial at all. In particular, brute-force algorithms do not work. Consider solving PLG for $n = 4$ and $\mathsf{D} = 3$ by exhaustive enumeration in a cube. The number of possibilities of choosing $\binom{n+\mathsf{D}}{n}$ points from the cube of $(n+1)^{\mathsf{D}}$ points is already 1.2×10^{31}! This number grows to 4.2×10^{81} for $n = 6$ and $\mathsf{D} = 3$! It is very challenging to *efficiently* find a poised lattice out of such an enormous number of candidates. After all, the PLG problem has to be solved for each irregular cell and the total cost of PLG should be a small fraction of that of solving the discretized equations.

Finally, a PLG algorithm must be able to handle all possible geometric configurations of an irregular boundary; the feasible set K in the signature of PLG serves this purpose. Note that phyiscally meaningful regions with arbitrarily complex topology and geometry can be accurately and efficiently represented by Yin sets [39]. We also insist that the algorithm operates in a dimension-agnostic manner so that there is no need to switch gears for different dimensions.

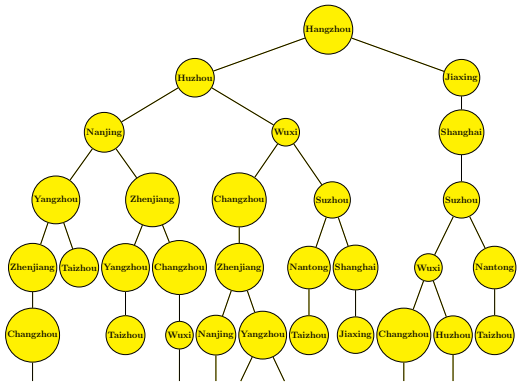
Given the above discussions on the nature of PLG, we believe that tools in traditional approximation theory are insufficient to satisfactorily solve the PLG problem. A coupling to search algorithms in artificial intelligence (AI) turns out to be helpful.

1.4 A bit of a mixture of approximation theory, group theory, and AI fused into one algorithm

A *search algorithm* is an AI algorithm for finding the best solution of a *search problem*, which consists of the *initial state*, the *goal state*, and a *state space*. A search algorithm solves the search problem by first organizing the state space into a special graph (typically a spanning tree) and then traversing the graph from the initial state by repeatedly determining the next best move according to some heuristic or deductive strategy. For an informed search problem, a solution is a path from the initial state to the goal state that minimizes certain cost function. Typical applications of search algorithms



(a) A map of a local region in east China. An integer represents the railway distance between two cities.



(b) The solution space is a spanning tree of the graph in subplot (a). We ignore any leaf node that forms a Hamiltonion circle with its parent nodes.

Figure 2: An example search problem of finding the shortest path from Hangzhou to Taizhou. The initial state and the goal state are Hangzhou and Taizhou, respectively. This search problem can be solved by search algorithms such as the depth-first search, the best-first search, and the A^* search. See [27, Chap. 3] for more details.

include pathfinding, optimization, and game playing. Figure 2 demonstrates how the shortest-path problem is solved by a search algorithm.

The PLG problem can be formulated as a search problem with its initial state as the singleton lattice $\{\mathbf{q}\}$ that only contains the starting point. There are, however, some key differences between PLG and a typical AI search problem such as that in Figure 2. As an obvious difference, there is only one goal state in the shortest path problem while there are many goal states in the PLG problem.

More importantly, the state space has different structures in these two cases. For the shortest path problem, the cost function, i.e., the total length of the traversed path, is always the sum of the distances of the constituting edges. This leads to a natural choice of organizing the state space into a spanning tree, by appending one city at a time to leaf nodes. The resulting structure of the state space is conducive to minimizing the cost function. In addition, visiting one city at a time appears to be the *only* appropriate way to construct the state space. In this sense, we say that the state space of the shortest path problem has a *static* structure.

In contrast, there is no intuitive way to construct the state space for the PLG problem, because it is difficult to convert the poisedness of lattices to the minimization of any cost functions: a lattice is either poised or not poised. Adding one point at a time into the lattice could be one way to construct

the state space, but it is by no means the only way. In fact, in multiple dimensions, it is more natural to add at a time $m_k := \binom{k+\mathsf{D}+1}{k+1} - \binom{k+\mathsf{D}}{k}$ points so that the polynomial degree of a non-initial state is greater by one than that of its parent state. Even for this strategy, one still has many possible choices of the m_k points to maintain the invariant of poisedness; furthermore, these choices depend on the particularity of the input feasible set K which is unknown at the time of algorithm design. In summary, the structure of the state space of the PLG problem is highly *dynamic*.

Our algorithm for PLG is a fusion of elementary concepts from approximation theory, group theory, and AI. First, we observe that only one poised lattice is needed per cell and there is no need to find all poised lattices in the feasible set K. Hence we focus on a particular class of poised lattices, namely the triangular lattices in Definition 3.1, to specialize the PLG problem to the problem of triangular lattice generation (TLG); see Definition 3.8. Second, for fixed n and D we study the structure of the group $\mathcal{X}$ of triangular lattices and show that $\mathcal{X}$ is isomorphic to the group of D-permutations that act on $\mathcal{X}$, c.f. Definition 3.12. Exploiting this isomorphism, we identify the state space of poised lattices with the state space of D-permutations. This conversion furnishes an organization of the state space via orbits of points under the D-permutations so that a triangular lattice can be obtained with optimal efficiency; see Theorem 4.7. Finally, we generate a poised lattice via traversing the state space by a depth-first search with backtracking.

The above TLG algorithm is fully explained in Sections 3 and 4.

1.5 Contributions of this work

The TLG algorithm solves the PLG problem and yields straightforward answers of questions posed in Section 1.1.

(A.1) Utilizing the TLG algorithm, we propose a new FD method, called the PLG-FD method, that retains Cartesian grids for comparable simplicity of legacy FD methods and solves PDEs on arbitrarily complex irregular domains with fourth-order accuracy. See Section 5 for more details.

(A.2) After generating the triangular lattice, we fit a complete multivariate polynomial and generate a discrete equation for each grid point according to the PDE and its boundary conditions. This process is completely independent of the specific form of the PDE. In particular, PDEs with cross derivatives and complex boundary conditions can be handled with ease.

(A.3) The TLG algorithm gives users of EB methods precise controls over least squares. Starting with the generated poised lattice, one can keep

adding nearby points until the condition number reaches a minimum; this results in an excellent balance between conditioning and efficiency.

(A.4) The algorithmic steps of the PLG-FD method are conceptually the same for different dimensions and different orders of accuracy, thus a single implementation covers a wide range of applications.

Apart from the above contributions for FD/FV methods, the TLG algorithm goes well with interpolation methods [29, 23, 8] that compute the interpolating polynomials from a given poised lattice. It might also be helpful to the design of search algorithms for AI applications. In particular, the strategy of dynamically organizing the state space by exploiting algebraic structures of the given problem could be very useful for an informed search where the information is not conducive to straightforward solutions.

The rest of this paper is organized as follows. In Section 2, we introduce notation and preliminary concepts. In Section 3, we examine the concept of triangular lattices, formalize the TLG problem, and analyze it from the viewpoint of group actions of permutations upon triangular lattices. In Section 4, we formalize the TLG algorithm as our solution to the PLG problem, based on which the new PLG-FD method is proposed in Section 5. In Section 6, we test the TLG algorithm as well as the PLG-FD method by performing a variety of numerical experiments, demonstrating fourth-order convergence of the PLG-FD method for elliptic problems on irregular domains. We conclude this work with some research prospects in Section 7.

2 Preliminaries

In this section, we introduce notation and collect relevant definitions and results from several distinct disciplines to make this paper self-contained.

2.1 Triangular lattices in the plane

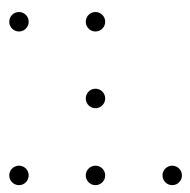
Most contents in this subsection are from Phillips [26, Chap. 5.2].

Definition 2.1. *A* triangular lattice of degree n in two dimensions *is a set of isolated points in* $\mathbb{R}^2$*,*

$$\mathcal{T}_n^2 = \{(x_i, y_j) : i, j \geq 0, i + j \leq n\}, \tag{6}$$

where x_i*'s are* $n + 1$ *distinct* x*-coordinates and* y_j*'s are* $n + 1$ *distinct* y*-coordinates.*

Example 2.2. *The constraints in (6) are not on the coordinates, but on their* indices. *Hence a triangular lattice might have a shape that does not look like a triangle.*



For example, the above triangular lattice

$$\mathcal{T}_2^2 = \{(0,0),(0,2),(1,0),(1,1),(1,2),(2,0)\}$$

has distinct coordinates $x_0 = 1$, $x_1 = 0$, $x_2 = 2$ *and* $y_0 = 0$, $y_1 = 2$, $y_2 = 1$.

Theorem 2.3. *A triangular lattice* $\mathcal{T}_n^2$ *is poised with respect to bivariate polynomials of degree no more than* n

$$\Phi_n^2 = \{1, x, y, x^2, xy, y^2, \ldots, x^n, x^{n-1}y, \ldots, xy^{n-1}, y^n\}; \tag{7}$$

the corresponding sample matrix M_2 *satisfies*

$$\det M_2 = C\psi_n(x)\psi_n(y), \tag{8}$$

where C *is a nonzero constant and* $\psi_n(x)$ *is a polynomial in terms of the* $n+1$ *distinct coordinates* x_i*'s,*

$$\psi_n(x) := \prod_{i=1}^{n}\prod_{\ell=0}^{i-1}(x_i - x_\ell)^{n+1-i}. \tag{9}$$

Proof. For any fixed i, ℓ with $i > \ell$, replacing (x_i, y_j) with (x_ℓ, y_j) in $\mathcal{T}_n^2$ makes the corresponding sample matrix singular for each $j = 0, 1, \ldots, n-i$; furthermore, j cannot exceed $n-i$ because (6) dictates $i + j \le n$. Therefore the number of this type of replacements is $n - i + 1$, and hence the exponent of $(x_i - x_\ell)$ in (9) is $n - i + 1$.

Now we vary ℓ while keeping i fixed. Since there are i indices less than i, the term $\prod_{\ell=0}^{i-1}(x_i - x_\ell)^{n+1-i}$ contributes to a total degree of $i(n+1-i)$ in terms of the $n+1$ coordinates $x_0, \ldots, x_n$. It follows that the total degree of $\psi_n(x)$ is

$$\sum_{i=1}^{n} i(n+1-i) = (n+1)\sum_{i=1}^{n} i - \sum_{i=1}^{n} i^2 = \frac{n(n+1)(n+2)}{6}, \tag{10}$$

where the second equality is proven by an easy induction.

Similarly, $\det M_2$ must contain a factor of $\psi_n(y)$, of which the total degree is also (10).

The determinant of the sample matrix M_2 in (2) is also a polynomial in terms of the variables $x_0, x_1, \ldots, x_n$ and $y_0, y_1, \ldots, y_n$, with each monomial being a product of all basis functions in (7) evaluated at some point

(x_i, y_j). Hence the total degree of $\det M_2$ in the variables $x_0, x_1, \ldots, x_n$ and $y_0, y_1, \ldots, y_n$ is

$$\sum_{i=1}^{n} i(i+1) = \frac{n(n+1)(n+2)}{3}, \tag{11}$$

where i refers to the degree of a monomial and $i+1$ the number of monomials of degree i, c.f. (7).

The proof is completed by the fact that if two complete polynomials have the same variables, the same total degree, and the same factors in terms of the same variables, then one is a constant multiple of the other. □

The kth *divided difference* of a function $f : \mathbb{R} \to \mathbb{R}$ over $x_0, x_1, \ldots, x_k \in \mathbb{R}$ is

$$[x_0, x_1, \ldots, x_k]f = \frac{[x_1, x_2, \ldots, x_k]f - [x_0, x_1, \ldots, x_{k-1}]f}{x_k - x_0} \tag{12}$$

with the recursion termination condition as $[x_0]f = f(x_0)$. We use the notation $[x_0, x_1, \ldots, x_k]f$ to emphasize the functional nature of divided differences.

The following is an error estimate for interpolation on triangular lattices.

Theorem 2.4. *For any scalar function f whose domain includes $\mathcal{T}_n^2$, we have,*

$$\forall m = 0, 1, \ldots, n, \quad f(x, y) = p_m(x, y) + r_m(x, y), \tag{13}$$

where $p_m(x, y)$ is a polynomial interpolator of $f(x, y)$ and $r_m(x, y)$ is the remainder on $\mathcal{T}_m^2$,

$$p_m(x, y) = \begin{cases} [x_0][y_0]f = f(x_0, y_0), & m = 0; \\ p_{m-1}(x, y) + q_m(x, y), & m > 0, \end{cases} \tag{14a}$$

$$q_m(x, y) = \sum_{k=0}^{m} \pi_k(x)\pi_{m-k}(y)[x_0, \ldots, x_k][y_0, \ldots, y_{m-k}]f, \tag{14b}$$

$$\begin{aligned} r_m(x, y) = &\sum_{k=0}^{m} \pi_{k+1}(x)\pi_{m-k}(y)[x, x_0, \ldots, x_k][y_0, \ldots, y_{m-k}]f \\ &+ \pi_{m+1}(y)[x][y, y_0, \ldots, y_m]f, \end{aligned} \tag{14c}$$

where the symmetric polynomial $\pi_n(x)$ is

$$\pi_n(x) = \begin{cases} 1, & n = 0; \\ \prod_{i=0}^{n}(x - x_i), & n > 0. \end{cases} \tag{15}$$

Proof. See [26, p. 180]. □

2.2 The symmetric group

A *group* is an algebra $(G, *, ^{-1}, e)$ where G is a set, $* : G \times G \to G$ a binary operation, and $^{-1} : G \to G$ a unitary operation such that the following axioms hold:

(GRP-1) associativity
$\forall a, b, c \in G, \quad (a * b) * c = a * (b * c)$;

(GRP-2) identity element
$\exists e \in G$, s.t. $\forall x \in G, \ e * x = x * e = x$;

(GRP-3) inverse element
$\forall a \in G, \exists a' \in G$ s.t. $a' * a = a * a' = e$;

H is a *subgroup* of G if $H \subset G$ and $(H, *, ^{-1}, e)$ is a group.

Definition 2.5. *A* permutation of a set A *is a bijective function* $\sigma : A \to A$.

Let A be a non-empty set, and let S_A be the collection of all permutations of A. Then $(S_A, \circ, ^{-1}, e)$ is a group where the identity e is the identity function.

Definition 2.6. *Let A be the finite set $\{1, 2, \ldots, n\}$. The group of all permutations of A is called the* symmetric group on n letters*, and is denoted by S_n or S_A.*

Cayley's theorem states that every group G is isomorphic to a subgroup of S_G.

Definition 2.7. *An* action of a group G on a set X *is a map $G \times X \to X$, written $(g, x) \mapsto g(x)$ for $g \in G$ and $x \in X$, that satisfies (i) $\forall x \in X$, $e(x) = x$ and (ii) $\forall x \in X$, $\forall g_1, g_2 \in G$, $(g_1 * g_2)(x) = g_1(g_2(x))$. X is called a* G-set *if G has an action on X. The* orbit of $x \in X$ under a group G acting on X *is the set $\{g(x) : g \in G\}$.*

2.3 Backtracking

Consider a task that consists of a sequence of k independent steps. The *fundamental principle of counting* states that the total number of distinct ways to complete the task is $\prod_{i=1}^{k} n_i$, where n_i is the number of different choices for the ith step. From another perspective, we can view the process of completing the task as traversing a directed tree. The root node of the tree represents the starting point with no steps done yet. The root node has n_1 children, each of which represents one way to finish the first step. Inductively, the n_i ways to finish the ith step are represented by n_i nodes, all of which are children of the single node representing the $(i-1)$th step. The $\prod_{i=1}^{k} n_i$ leaf nodes form the solution space of all different ways to complete the task. This tree is called the *spanning tree* of the solution space.

Definition 2.8 (Backtracking)**.** *Let P be a problem that admits an incremental assemblage of the solution and hence a spanning-tree organization of the solution space.* Backtracking *solves P by calling* `BackTrack`$(P,$ `root`$(P), \{\})$,

Procedure BackTrack(P, $\mathbf{r}$, T)

Input: $\mathbf{r}$ *is a starting node in the solution space and T is a set of solutions*

Side effects **:** *T contains all valid solutions*

```
1  if accept(P, r) then
2  |  T = T ∪ {r}
3  |  if stopAfterAccept(P, r) then return
4  else if reject(P, r) then
5  |  return
6  end
7  s ← first(P, r)
8  while s is not null do
9  |  BackTrack(P, s, T)
10 |  s ← next(P, r, s)
11 end
```

where the following subroutines are user-defined according to the specific problem at hand,

- `root`(P)*: return the root node of the spanning tree of the solution space,*
- `accept`$(P, \mathbf{r})$*: return true if and only if* $\mathbf{r}$ *is already a solution; return false otherwise,*
- `stopAfterAccept`$(P, \mathbf{r})$*: return true if and only if the valid solution* $\mathbf{r}$ *can never be extended to another valid solution; return false otherwise,*
- `reject`$(P, \mathbf{r})$*: return true if and only if the partial candidate (or node)* $\mathbf{r}$ *can never be completed to a valid solution; return false otherwise,*
- `first`$(P, \mathbf{r})$*: return the first extension of* $\mathbf{r}$ *in the spanning tree if* $\mathbf{r}$ *is extendable; return null otherwise,*
- `next`$(P, \mathbf{r}, \mathbf{s})$*: return the next extension of* $\mathbf{r}$ *after* $\mathbf{s}$ *if* $\mathbf{r}$ *is still extendable; return null otherwise.*

Backtracking incrementally builds solutions by abandoning any candidate that cannot possibly be completed to a valid solution. Hence it can be much faster than brute-force enumeration since the procedure `reject` may eliminate a large number of candidates with a single test. In particular, if `reject` always returns false, the procedure in Definition 2.8 reduces to a brute-force enumeration.

In Definition 2.8, although results of the two subroutines `accept`$(P, \mathbf{r})$ and `reject`$(P, \mathbf{r})$ cannot be both `true`, they may be both `false`: an intermediate

node $\mathbf{r}$ for which both subroutines return `false` may later be extended to a valid solution. On the other hand, if `accept`$(P, \mathbf{r})$ is true for some $\mathbf{r}$, only two cases are possible: either $\mathbf{r}$ may be extended to another valid solution, or, $\mathbf{r}$ can never be so. In the latter case `stopAfterAccept` should return true.

All non-null nodes generated by `first`$(P, \mathbf{r})$ and `next`$(P, \mathbf{r}, \mathbf{s})$ have the same rank (or *level*) with respect to `root`(P), i.e. the number of extensions from `root`(P).

3 Analysis and algebra

3.1 Triangular lattices in D dimensions

Hereafter we denote the first $n+1$ nonnegative integers and the first n positive integers respectively by

$$\mathbb{Z}_n := \{0, 1, \ldots, n\}, \quad \mathbb{Z}_n^+ := \{1, \ldots, n\}. \tag{16}$$

Definition 3.1. *A subset $\mathcal{T}_n^D$ of $\mathbb{R}^D$ is called a* triangular lattice of degree n in D dimensions *if for each of the D dimensions there exist $n+1$ distinct coordinates and a numbering of these coordinates,*

$$\begin{bmatrix} p_{1,0} & p_{1,1} & \cdots & p_{1,n} \\ p_{2,0} & p_{2,1} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{D,0} & p_{D,1} & \cdots & p_{D,n} \end{bmatrix} \in \mathbb{R}^{D\times(n+1)}, \tag{17}$$

such that $\mathcal{T}_n^D$ can be expressed as

$$\mathcal{T}_n^D = \left\{ (p_{1,k_1}, p_{2,k_2}, \ldots, p_{D,k_D}) \in \mathbb{R}^D : \ k_i \in \mathbb{Z}_n; \ \sum_{i=1}^{D} k_i \le n \right\}, \tag{18}$$

where $p_{i,j}$ denotes the jth coordinate of the ith variable p_i.

Hereafter, we will reserve the symbol i for indexing dimensions: $i = 1, 2, \ldots,$ D.

Example 3.2. *For D $= 2$, Definition 3.1 reduces to Definition 2.1 since (18) simplifies to*

$$\mathcal{T}_n^2 = \{(p_{1,k_1}, p_{2,k_2}) : \ k_1, k_2 \ge 0; \ k_1 + k_2 \le n\},$$

which is the same as (6).

Lemma 3.3. *The cardinality of a triangular lattice is*

$$\#\mathcal{T}_n^{D} = \binom{n+\mathsf{D}}{\mathsf{D}} = \sum_{i=0}^{n} \#\mathcal{T}_{=i}^{D} = \sum_{i=0}^{n} \binom{i+\mathsf{D}-1}{\mathsf{D}-1}, \tag{19}$$

where

$$\mathcal{T}_{=m}^{D} := \left\{ (p_{1,k_1}, p_{2,k_2}, \ldots, p_{D,k_D}) : k_i \geq 0; \sum_{i=1}^{D} k_i = m \right\} \tag{20}$$

and its cardinality is

$$\#\mathcal{T}_{=m}^{D} = \#\mathcal{T}_m^{D} - \#\mathcal{T}_{m-1}^{D} = \binom{m+\mathsf{D}-1}{\mathsf{D}-1}. \tag{21}$$

Proof. As the only nontrivial constraint on $\mathcal{T}_n^{\mathsf{D}}$ in (18), the sum of the D nonnegative integers $i_1, \ldots, i_{\mathsf{D}}$ cannot exceed n. Hence $\#\mathcal{T}_n^{\mathsf{D}}$ equals the number of possibilities of placing n indistinguishable balls into $\mathsf{D}+1$ distinguishable urns, with the first D urns corresponding to the D dimensions, respectively, and the last urn accounting for the deficit of $\sum_k i_k$ from n. This ball-urn problem is further equivalent to choosing D balls from $n+\mathsf{D}$ balls in a row because the chosen D balls divide the rest n balls into $\mathsf{D}+1$ consecutive arrays of balls, each of which corresponds to an urn. This proves the first equality in (19).

As for the second equality in (19), (18) and (20) imply $\mathcal{T}_n^{\mathsf{D}} = \cup_{m=0}^{n} \mathcal{T}_{=m}^{\mathsf{D}}$ and that two subsets $\mathcal{T}_{=m}^{\mathsf{D}}$ and $\mathcal{T}_{=n}^{\mathsf{D}}$ are disjoint if and only if $m \neq n$.

An easy induction shows (21), which further implies the third equality in (19). □

Definition 3.4. *The* D-variate polynomials of degree no more than n *form the set*

$$\Phi_n^{D} = \left\{ p_1^{e_1} p_2^{e_2} \ldots p_D^{e_D} : e_i \geq 0; \sum_{i=1}^{D} e_i \leq n \right\}, \tag{22}$$

and the D-variate polynomials of degree m *form the set*

$$\Phi_{=m}^{D} = \left\{ p_1^{e_1} p_2^{e_2} \ldots p_D^{e_D} : e_i \geq 0; \sum_{i=1}^{D} e_i = m \right\}. \tag{23}$$

Lemma 3.5. *The results on triangular lattices in Lemma 3.3 also hold for sets of multivariate polynomials. More precisely, (19) and (21) still hold when the symbol* $\mathcal{T}$ *is replaced with the symbol* Φ.

Proof. This follows from a natural bijection between Φ_n^{D} and $\mathcal{T}_n^{\mathsf{D}}$, the restriction of which is also a bijection between $\Phi_{=n}^{\mathsf{D}}$ and $\mathcal{T}_{=n}^{\mathsf{D}}$. □

Lemma 3.6. *For any positive integers* D *and* n*, we have*

$$(\mathsf{D}+1)\sum_{j=1}^{n} j\binom{n-j+\mathsf{D}}{\mathsf{D}} = \sum_{j=1}^{n} j\binom{j+\mathsf{D}}{\mathsf{D}}. \tag{24}$$

Proof. For $n=1$, both sides reduce to $\mathsf{D}+1$. Suppose (24) holds. Then the inductive step also holds because

$$\begin{aligned}(\mathsf{D}+1)\sum_{j=1}^{n+1} j\binom{n+1-j+\mathsf{D}}{\mathsf{D}} =& (\mathsf{D}+1)\sum_{i=0}^{n}(i+1)\binom{n-i+\mathsf{D}}{\mathsf{D}} \\ =& (\mathsf{D}+1)\sum_{i=0}^{n} i\binom{n-i+\mathsf{D}}{\mathsf{D}} + (\mathsf{D}+1)\sum_{j=0}^{n}\binom{j+\mathsf{D}}{\mathsf{D}} \\ =& \sum_{j=1}^{n} j\binom{j+\mathsf{D}}{\mathsf{D}} + (\mathsf{D}+1)\binom{n+\mathsf{D}+1}{\mathsf{D}+1} \\ =& \sum_{j=1}^{n+1} j\binom{j+\mathsf{D}}{\mathsf{D}},\end{aligned}$$

where the first two steps follow from variable substitutions, the third step from the induction hypothesis and the well-known identity $\sum_{j=0}^{n}\binom{\mathsf{D}+j}{\mathsf{D}} = \binom{\mathsf{D}+n+1}{\mathsf{D}+1}$, and the last step from

$$(\mathsf{D}+1)\binom{n+\mathsf{D}+1}{\mathsf{D}+1} = (\mathsf{D}+1)\frac{(n+\mathsf{D}+1)!}{(\mathsf{D}+1)!n!} = (n+1)\frac{(n+\mathsf{D}+1)!}{\mathsf{D}!(n+1)!} = (n+1)\binom{n+\mathsf{D}+1}{\mathsf{D}}.$$

□

Theorem 3.7. *A triangular lattice* $\mathcal{T}_n^{\mathsf{D}}$ *is poised with respect to the* D*-variate polynomials of degree no more than* n*; the corresponding sample matrix* M_{D} *satisfies*

$$\det M_{\mathsf{D}} = C\prod_{k=1}^{\mathsf{D}} \psi_n(p_k) \tag{25}$$

where C *is a nonzero constant and* $\psi_n(p_k)$ *is a polynomial in terms of the* $n+1$ *distinct coordinates of the variable* p_k*,*

$$\psi_n(p_k) := \prod_{i_k=1}^{n}\prod_{\ell=0}^{i_k-1}(p_{k,i_k}-p_{k,\ell})^{\alpha(i_k)}; \quad \alpha(i_k) := \binom{n-i_k+\mathsf{D}-1}{\mathsf{D}-1}. \tag{26}$$

Proof. We follow the strategy in the proof of Theorem 2.3, with more complicated book-keeping on the combinatorics.

Consider the kth variable p_k. For any fixed i_k and ℓ with $i_k > \ell$, replacing the coordinate p_{k,i_k} with $p_{k,\ell}$ in a point

$$\mathbf{p}_k := (p_{1,i_1}, \ldots, p_{k,i_k}, \ldots, p_{\mathsf{D},i_\mathsf{D}}) \in \mathcal{T}_n^\mathsf{D} \tag{27}$$

makes the corresponding sample matrix M_D singular. Furthermore, when the coordinate index of the kth variable p_k is fixed at i_k in $\mathcal{T}_n^\mathsf{D}$, the cardinality of $\mathcal{T}_n^\mathsf{D}$ reduces to $\#\mathcal{T}_{n-i_k}^{\mathsf{D}-1}$, which, by Lemma 3.3, must equal $\alpha(i_k)$. In other words,

$$\#\left\{(p_{1,i_1}, \ldots, p_{k,i_k}, \ldots, p_{\mathsf{D},i_\mathsf{D}}) \in \mathcal{T}_n^\mathsf{D} : \ i_k \text{ is fixed}\right\}$$

equals the cardinality of a triangular lattice of degree $n-i_k$ in $\mathsf{D}-1$ dimensions because an index of i_k has been consumed from the total index n and one of the D dimensions has already been fixed. Therefore, the number of possible $\mathbf{p}_k$'s that admit the replacement of p_{k,i_k} with $p_{k,\ell}$ is $\alpha(i_k)$, and this justifies the exponent of $(p_{k,i_k} - p_{k,\ell})$ in (26).

Now we vary ℓ while keeping i_k fixed. Since there are i_k indices less than i_k, the term $\prod_{\ell=0}^{i_k-1}(p_{k,i_k} - p_{k,\ell})^{\alpha(i_k)}$ contributes to a total degree $i_k\alpha(i_k)$ in terms of the $n+1$ coordinates $p_{k,0}, \ldots, p_{k,n}$. It follows that $\psi_n(p_k)$ must be a factor of $\det M_\mathsf{D}$ and the total degree of $\psi_n(p_k)$ is

$$\sum_{i_k=1}^{n} i_k\alpha(i_k) = \sum_{j=1}^{n} j\binom{n-j+\mathsf{D}-1}{\mathsf{D}-1}.$$

Similarly, $\det M_\mathsf{D}$ must contain a factor of $\psi_n(p_j)$ for each variable p_j, $j = 1, \ldots, \mathsf{D}$. Hence the total degree of $\det M_\mathsf{D}$ is at least

$$\xi := \mathsf{D}\sum_{j=1}^{n} j\binom{n-j+\mathsf{D}-1}{\mathsf{D}-1}.$$

The determinant of the sample matrix M_D is also a polynomial in terms of the coordinates $p_{1,0}, p_{1,1}, \ldots, p_{1,n}, \ldots, p_{\mathsf{D},0}, p_{\mathsf{D},1}, \ldots, p_{\mathsf{D},n}$, with each monomial being a product of all basis functions in (22) evaluated at some point $(p_{1,i_1}, p_{2,i_2}, \ldots, p_{\mathsf{D},i_\mathsf{D}})$. By Lemma 3.5 and (21), the total degree of $\det M_\mathsf{D}$ equals

$$\eta := \sum_{i=1}^{n} i\binom{i+\mathsf{D}-1}{\mathsf{D}-1},$$

where i refers to the degree of monomials in the subset $\Phi_{=i}^\mathsf{D}$ and $\binom{i+\mathsf{D}-1}{\mathsf{D}-1}$ the cardinality of $\Phi_{=i}^\mathsf{D}$.

Lemma 3.6 implies $\xi = \eta$. Hence the terms in $\prod_{k=1}^{\mathsf{D}} \psi_n(p_k)$ constitute all the non-constant factors of $\det M_\mathsf{D}$, which yields (25). □

3.2 The problem of triangular lattice generation (TLG)

In Definition 3.1, a triangular lattice is a set of isolated points in $\mathbb{R}^{\mathsf{D}}$ such that projecting these points to any axis yields $n+1$ coordinates. Restricting $\mathbb{R}^{\mathsf{D}}$ to $\mathbb{Z}^{\mathsf{D}}$ does not lose any generality since each triangular lattice in $\mathbb{R}^{\mathsf{D}}$ is isomorphic to some triangular lattice in $\mathbb{Z}^{\mathsf{D}}$.

Definition 3.8 (The TLG problem)**.** *Denote the* D-dimensional cube of size $n+1$ *as*

$$\mathbb{Z}_n^D := (\mathbb{Z}_n)^D = \{0, 1, \ldots, n\}^D \tag{28}$$

and define the set of all triangular lattices of degree n *in* $\mathbb{Z}_n^D$ *as*

$$\mathcal{X} := \left\{\mathcal{T}_n^D : \mathcal{T}_n^D \subset \mathbb{Z}_n^D\right\}. \tag{29}$$

For a set of feasible nodes $K \subseteq \mathbb{Z}_n^D$ *and a starting point* $\mathbf{q} \in K$*, the* triangular-lattice generation (TLG) problem *seeks* $\mathcal{T} \in \mathcal{X}$ *such that* $\mathbf{q} \in \mathcal{T}$ *and* $\mathcal{T} \subseteq K$*.*

The above formulation simplifies the lattice selection problem in Definition 1.2 in that (i) the desirable poised lattice is restricted to the type of triangular lattices, and (ii) only the *numbering* of the coordinates in $\mathbb{Z}_n^{\mathsf{D}}$ needs to be determined.

3.3 A group action of D-permutations on triangular lattices

Definition 3.9. *The* principal lattice *of degree* n *in* $\mathbb{Z}^D$ *is*

$$\mathcal{P}_n^D = \left\{(j_1, \ldots, j_D) \in \mathbb{Z}^D : \forall k = 1, 2, \cdots, D,\ j_k \geq 0; \sum_{k=1}^{D} j_k \leq n\right\}. \tag{30}$$

As an example, the principal lattice of degree 2 in two dimensions is

$$\mathcal{P}_2^2 = \{(0,0), (0,1), (0,2), (1,0), (1,1), (2,0)\}. \tag{31}$$

Definition 3.10. *The* m-slice of a subset $T \subseteq \mathbb{Z}_n^D$ *across the* i*th dimension is a subset of* T *defined as*

$$L_{i,m}(T) = \{\mathbf{p} \in T : p_i = m\}. \tag{32}$$

As in Definition 3.1, the $n+1$ coordinates $0, 1, \ldots, n$ in $\mathbb{Z}_n$ are given an ordering for each dimension i and $p_{i,m}$ represents the mth coordinates along the ith dimension. In particular, $p_{i,m}$ may or may not equal m.

Lemma 3.11. *Any* $p_{i,m}$*-slice of a triangular lattice of degree* n *in* D *dimensions is a triangular lattice of degree* $n-m$ *in* $\mathsf{D}-1$ *dimensions.*

Proof. By Definition 3.1, the triangular lattice is

$$\mathcal{T}_n^{\mathsf{D}} = \left\{ (p_{1,j_1}, p_{2,j_2}, \ldots, p_{D,j_D}) : \ j_k \geq 0; \ \sum_{k=1}^{\mathsf{D}} j_k \leq n \right\},$$

where each variable p_i has exactly $n+1$ coordinates. By Definition 3.10, we have

$$L_{i,p_{i,m}}(\mathcal{T}) = \left\{ \left(p_{1,j_1}, \ \ldots, \ p_{i-1,j_{i-1}}, \ p_{i,m}, \ p_{i+1,j_{i+1}}, \ \ldots, \ p_{D,j_D}\right) : \ j_k \geq 0; \ \sum_{k\neq i,k=1}^{\mathsf{D}} j_k \leq n-m \right\},$$

which, by Definition 3.1, is a triangular lattice of degree $n-m$ in $\mathsf{D}-1$ dimensions after renumbering the $n-m+1$ coordinates for each dimension $k \neq i$. □

Definition 3.12. *A* D-permutation of degree n, *written*

$$A = (a_1, a_2, \ldots, a_D)^T,$$

is a map $A : \mathbb{Z}_n^D \to \mathbb{Z}_n^D$ *defined as*

$$A\mathbf{p} = \left(a_1(p_1), a_2(p_2), \cdots, a_D(p_D)\right)^T, \tag{33}$$

where each $a_i : \mathbb{Z}_n \to \mathbb{Z}_n$ *is a permutation.*

Notation 1. *Denote by* G_n^D *the* set of all D-permutations of degree n.

Definition 3.13. *The* multiplication of two D-permutations *is a binary operation* $\cdot : G_n^D \times G_n^D \to G_n^D$ *given by*

$$A \cdot B = (a_1 \circ b_1, a_2 \circ b_2, \ldots, a_D \circ b_D)^T, \tag{34}$$

where "$\circ$*" denotes function composition.*

Definition 3.14. *The* inverse of a D-permutation A *is a unitary operation* $^{-1} : G_n^D \to G_n^D$ *such that* A^{-1} *satisfies*

$$A^{-1} \cdot A = A \cdot A^{-1} = E = (e_1, e_2, \ldots, e_D)^T, \tag{35}$$

where E *denotes the distinguished* D*-permutation with each constituting permutation* e_i *as the identity map on* $\mathbb{Z}_n$.

Definition 3.14 is well defined because of (34) and the condition that each constituting permutation is a bijection.

Lemma 3.15. *The following algebra is a group,*

$$(G_n^D, \cdot, ^{-1}, E). \tag{36}$$

Proof. It is straightforward to verify the conditions of a group from Definitions 3.12, 3.13, and 3.14. □

Lemma 3.16. *For any $A \in G_n^{\mathsf{D}}$, the map σ_A given by*

$$\forall \mathcal{T} \in \mathcal{X},\ \sigma_A(\mathcal{T}) = A\mathcal{T} := \left\{A\mathbf{p} : \mathbf{p} \in \mathcal{T}\right\} \tag{37}$$

is a permutation of $\mathcal{X}$. In other words, D-permutations map triangular lattices to triangular lattices.

Proof. Since $\mathcal{T}$ is a triangular lattice, we know from Definition 3.1 that there exist $n+1$ distinct coordinates for each dimension such that *indices* of the D constituting coordinates of each point $\mathbf{p} \in \mathcal{T}$ add up to no more than n. The action of A upon $\mathcal{T}$ in (33) can be undone by applying A^{-1}; this means that for $A\mathcal{T}$ there exists a renumbering (specified by A^{-1}) of the coordinates along each axis such that $A\mathcal{T}$ is a triangular lattice. □

Lemma 3.17. *The set of triangular lattices $\mathcal{X}$ is a G_n^{D}-set with its group action $G_n^{\mathsf{D}} \times \mathcal{X} \to \mathcal{X}$ given by $\sigma_A(\mathcal{T})$ in (37).*

Proof. By Lemma 3.16, $\bullet(A, \mathcal{T}) = \sigma_A(\mathcal{T})$ indeed has the signature $G_n^{\mathsf{D}} \times \mathcal{X} \to \mathcal{X}$. By (35), we have

$$\forall \mathcal{T} \in \mathcal{X}, \quad E\mathcal{T} = \mathcal{T}.$$

In addition, for any $A, B \in G_n^{\mathsf{D}}$ and any $\mathcal{T} \in \mathcal{X}$, we have

$$\begin{aligned}(A \cdot B)\mathcal{T} &= \left\{(A \cdot B)\mathbf{p} : \mathbf{p} \in \mathcal{T}\right\} = \left\{((a_1 \circ b_1)(p_1), \ldots, (a_{\mathsf{D}} \circ b_{\mathsf{D}})(p_{\mathsf{D}}))^T : \mathbf{p} \in \mathcal{T}\right\} \\ &= \left\{(a_1(b_1(p_1)), \ldots, a_{\mathsf{D}}(b_{\mathsf{D}}(p_{\mathsf{D}})))^T : B\mathbf{p} \in \mathcal{T}\right\} = A(B\mathcal{T}),\end{aligned}$$

where the first step follows from (37), the second from (33) and (34), the third from Lemma 3.16, and the last from (37). Definition 2.7 completes the proof. □

Definition 3.18. *The* restoration of a triangular lattice $\mathcal{T}_n^{\mathsf{D}}$ *is a D-permutation* $R_{\mathcal{T}} = (r_1, r_2, \ldots, r_{\mathsf{D}})^T$ *such that*

$$\forall i = 1, 2, \ldots, \mathsf{D},\ \forall m \in \mathbb{Z}_n,\ r_i(m) = \#\left\{j \in \mathbb{Z}_n : \#L_{i,j}(\mathcal{T}) > \#L_{i,m}(\mathcal{T})\right\}. \tag{38}$$

(38) is well-defined as we shall see immediately that no two slices of a triangular lattice have the same cardinality.

Lemma 3.19. *The image of a triangular lattice $\mathcal{T}_n^{\mathsf{D}}$ under its restoration is the principal lattice $\mathcal{P}_n^{\mathsf{D}}$, i.e.*

$$R_{\mathcal{T}_n^{\mathsf{D}}}\mathcal{T}_n^{\mathsf{D}} = \mathcal{P}_n^{\mathsf{D}}. \tag{39}$$

Proof. Lemma 3.11 and Lemma 3.3 imply that the slices of $\mathcal{T}$ along each axis have pairwise distinct cardinalities. By Definition 3.18, the cardinalities of rearranged slices along the ith dimension are not changed by any constituting permutations except r_i. By Lemma 3.16, $R_{\mathcal{T}}\mathcal{T}$ is also a triangular lattice. Furthermore, cardinalities of the m-slices of $R_{\mathcal{T}}\mathcal{T}$ decrease strictly monotonically as m increases. There is only one such triangular lattice, namely $\mathcal{P}_n^{\mathsf{D}}$ in (30). Finally, $R_{\mathcal{T}}$ is a bijection because each constituting permutation is a bijection. □

As a D-permutation, a restoration is a bijection. and hence it has an inverse.

Definition 3.20. *The* formation of a triangular lattice $\mathcal{T}$ *is the inverse of its restoration, i.e.,*

$$A_{\mathcal{T}} = R_{\mathcal{T}}^{-1}. \tag{40}$$

Lemma 3.21. *The formation of a triangular lattice $\mathcal{T}_n^D$ maps the principal lattice $\mathcal{P}_n^D$ to $\mathcal{T}_n^D$,*

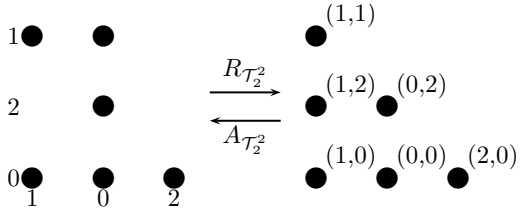
$$\mathcal{T}_n^D = A_{\mathcal{T}_n^D}\mathcal{P}_n^D. \tag{41}$$

Proof. This follows directly from Definitions 3.18 and 3.20 and Lemma 3.19. □

Example 3.22. *For the triangular lattice $\mathcal{T}_2^2$ in Example 2.2, its formation $A_{\mathcal{T}_2^2}$ equals its restoration $R_{\mathcal{T}_2^2}$,*

$$\begin{cases} r_1 : 0 \mapsto 1; 1 \mapsto 0; 2 \mapsto 2, \\ r_2 : 0 \mapsto 0; 1 \mapsto 2; 2 \mapsto 1. \end{cases} \tag{42}$$

The processes of restoration and formation are shown below.



On the left, the integers below the lattice are $r_1([0,1,2])$, i.e., the numbers of vertical slices whose cardinalities are larger than the current slice; the integers to the left of the lattice are $r_2([0,1,2])$, i.e., the numbers of horizontal slices whose cardinalities are larger than the current slice. On the right, the multiindex close to a dot is the preimage of the dot under $R_{\mathcal{T}_2^2}$, whose behavior is summarized as follows.

$R\mathbf{p} \in \mathcal{P}_2^2$	*(0,0)*	*(0,1)*	*(0,2)*	*(1,0)*	*(1,1)*	*(2,0)*
$\mathbf{p} \in \mathcal{T}_2^2$	*(1,0)*	*(1,2)*	*(1,1)*	*(0,0)*	*(0,2)*	*(2,0)*

Lemma 3.23. *Let $\mathcal{P}$ be the principal lattice of $\mathcal{X}$. The map $\varphi : G_n^{\mathsf{D}} \to \mathcal{X}$ given by $\varphi(A_{\mathcal{T}}) := A_{\mathcal{T}}\mathcal{P} = \mathcal{T}$ is bijective.*

Proof. For any $\mathcal{T} \in \mathcal{X}$, its restoration is uniquely defined in (38) and the formation of $\mathcal{T}$ always satisfies $A_{\mathcal{T}}\mathcal{P} = \mathcal{T}$. Hence φ is surjective. By Definition 3.12, we have $A_1\mathcal{P} \neq A_2\mathcal{P}$ for two D-permutations $A_1 \neq A_2$; hence φ is injective. □

Theorem 3.24. *The group $(G_n^{\mathsf{D}}, \circ)$ is isomorphic to $(\mathcal{X}, *)$ with the binary operation $*$ on $\mathcal{X}$ given by*

$$\mathcal{T} * \mathcal{S} := \varphi(\varphi^{-1}(\mathcal{T}) \circ \varphi^{-1}(\mathcal{S})). \tag{43}$$

Proof. This follows from (43) and Lemma 3.23. □

Corollary 3.25. *Let $S_{\mathcal{X}}$ be the symmetric group on $\mathcal{X}$. The map $\phi : G_n^{\mathsf{D}} \to S_{\mathcal{X}}$ given by*

$$\phi(A) = \sigma_A \tag{44}$$

is a monomorphism.

Proof. Consider the chain of maps

$$\phi : G_n^D \xrightarrow{\varphi} \mathcal{X} \xrightarrow{c} \mathcal{C}_{\mathcal{X}} \xhookrightarrow{i} \mathcal{S}_{\mathcal{X}}, \tag{45}$$

where $\mathcal{C}_{\mathcal{X}}$ is the subgroup of $\mathcal{S}_{\mathcal{X}}$ isomorphic to $\mathcal{X}$, c.f. Cayley's theorem, c is the corresponding isomorphism, and i is the inclusion. Then ϕ as a composition of the three homomorphisms in (45) is clearly a monomorphism because the first two are isomorphisms and the last one is a monomorphism. ϕ is not surjective since

$$\#G_n^D = (n+1)^{\mathsf{D}} < \#\mathcal{S}_{\mathcal{X}} = \big((n+1)^{\mathsf{D}}\big)!.$$

□

Corollary 3.26. *Each preimage-image pair of triangular lattices uniquely determines a D-permutation $A \in G_n^{\mathsf{D}}$, and consequently the corresponding $\sigma_A \in \mathcal{S}_{\mathcal{X}}$ as in (44) is also uniquely determined.*

Proof. If $A(\mathcal{T}_1) = \mathcal{T}_2$, set $A' = \varphi^{-1}(\mathcal{T}_2) \circ [\varphi^{-1}(\mathcal{T}_1)]^{-1}$ and we have $A' = A$. Here φ is given in Lemma 3.23, φ^{-1} is the inverse of φ, and the last $^{-1}$ is the inverse of D-permutations in Definition 3.14. Finally, (45) implies $\phi(A) \in \mathcal{C}_{\mathcal{X}}$ and that both φ and c are isomorphisms. □

Corollary 3.27. *The TLG problem $(K, \mathbf{q})$ in Definition 3.8 is solved by a triangular lattice $\mathcal{T}$ if and only if its formation $A_{\mathcal{T}}$ satisfies*

$$\exists \mathbf{p} \in \mathcal{P}_n^{\mathsf{D}},\ \textit{s.t.}\ A_{\mathcal{T}}\mathbf{p} = \mathbf{q}; \tag{46a}$$

$$\forall \mathbf{p} \in \mathcal{P}_n^{\mathsf{D}},\ A_{\mathcal{T}}\mathbf{p} \in K. \tag{46b}$$

Furthermore, all solutions to the TLG problems can be obtained by enumerating the formations.

Proof. The first statement follows from Definition 3.8, Lemma 3.19, and Definition 3.20 while the second statement follows from the isomorphism in Theorem 3.24. □

The significance of Corollary 3.27 is that, instead of enumerating all triangular lattices in $\mathcal{X}$, we can obtain *all* solutions to the TLG problem by enumerating the D-permutations in G_n^{D}, whose simple structure is very amenable to the enumeration.

Notation 2. *In matrix notation, a* D*-permutation is*

$$A = \begin{bmatrix} a_1(0) & a_1(1) & \dots & a_1(n) \\ a_2(0) & a_2(1) & \dots & a_2(n) \\ \vdots & \vdots & \ddots & \vdots \\ a_D(0) & a_D(1) & \dots & a_D(n) \end{bmatrix}, \tag{47}$$

where the row index i of the matrix starts from 1 while the column index j starts from 0. In other words, the (i, j)th-element of A is

$$A(i, j) = a_i(j). \tag{48}$$

3.4 Partitioning the principal lattice and D-permutations

When enumerating D-permutations and checking them against (46), there are two extremes. For a given D-permutation, we could check the $\binom{n+\mathsf{D}}{\mathsf{D}} = \frac{(n+\mathsf{D})!}{n!\mathsf{D}!}$ conditions one at a time: we deny the D-permutation upon the first time it fails any check; otherwise we retain it as a solution. On the other hand, we could check all conditions in (46) *simultaneously*. For example, we could represent a subset P of $\mathbb{Z}_n^{\mathsf{D}}$ by a binary number, each bit of which corresponds to a point $p \in \mathbb{Z}_n^{\mathsf{D}}$, with 1 meaning $p \in P$ and 0 meaning $p \notin P$; then checking all conditions in (46) reduces to a single calculation of the logical conjunction of two binary numbers that represent K and the triangular lattice $A_{\mathcal{T}}\mathcal{P}$.

In the former method the number of steps $\frac{(n+\mathsf{D})!}{n!\mathsf{D}!}$ is unnecessarily large while in the latter method much computation is repeated for similar D-permutations. Following the discussion after Corollary 3.27, we group the points of a principal lattice into $\mathsf{D}(n+1)$ equivalence classes, and at each stage we check (46) for points in a single equivalence class. This helps to preclude at early stages those D-permutations that do not satisfy (46b).

Definition 3.28. *A* test set $W_{(\ell,m)}$ *is an equivalence class of points in the principal lattice,*

$$\bigcup_{(\ell,m)\in\mathbb{Z}_D^+\times\mathbb{Z}_n} W_{(\ell,m)} = \mathcal{P}_n^D. \tag{49}$$

Recall that a *partial function* from Y to Z is a function $Y' \to Z$ on some $Y' \subseteq Y$.

Definition 3.29. *The* (ℓ, m)-partial D-permutation*, denoted by* $A^{(\ell,m)}$*, is a partial function on the test set* $W_{(\ell,m)}$ *that satisfies*

$$\forall \mathbf{p} \in W_{(\ell,m)},\ A^{(\ell,m)}\mathbf{p} = A\mathbf{p}. \tag{50}$$

Although $A^{(\ell,m)} \neq A$, their actions on $W_{(\ell,m)}$ are exactly the same. Instead of enumerating all $A \in G_n^{\mathsf{D}}$, we enumerate all $A^{(\ell,m)}$'s by adding one number at a time to the D-by-$(n+1)$ grid in (47), so that they are naturally organized into a spanning tree. The parent-child relation in this spanning tree and the incremental construction of D-permutations can be made precise by a linear ordering of pairs such as the one below.

Definition 3.30. *The* linear ordering of integer pairs *on a grid* $\mathbb{Z}_D^+ \times \mathbb{Z}_n$ *is the column-wise ordering of the grid, i.e., the total ordering obtained by stacking all the columns of the grid into one column. More precisely, this ordering is a bijection* s *that maps a pair* $(i, j) \in \mathbb{Z}_D^+ \times \mathbb{Z}_n$ *to a scalar index* $k \in \mathbb{Z}^+_{D(n+1)}$*,*

$$s(i, j) = i + j\,\mathsf{D}; \tag{51}$$

$$s^{-1}(k) = \left(1 + (k-1)\ mod\ \mathsf{D},\ \left\lfloor \frac{k-1}{D} \right\rfloor\right), \tag{52}$$

where $\lfloor \cdot \rfloor : \mathbb{Q} \to \mathbb{N}$ *is the floor operator.*

The matrix of a partial D-permutation is simply

$$A^{(\ell,m)}(i, j) = \begin{cases} A(i, j) & \text{if } s(i, j) \leq s(\ell, m); \\ -1 & \text{otherwise,} \end{cases} \tag{53}$$

where s is defined in (51), and "-1" indicates undefined behavior. Since s is a bijection, it also makes sense to write

$$\forall t = s(\ell, m),\ A^{(t)} := A^{(\ell,m)}. \tag{54}$$

Lemma 3.31. *A triangular lattice* $\mathcal{T}_n^D$ *has the partition*

$$\mathcal{T}_n^D = \bigcup_{(\ell,m)\in\mathbb{Z}_D^+\times\mathbb{Z}_n} A_{\mathcal{T}}^{(\ell,m)} W_{(\ell,m)}, \tag{55}$$

where the terms $A_{\mathcal{T}}^{(\ell,m)} W_{(\ell,m)}$ *are pairwise disjoint.*

Proof. By Definitions 3.28 and 3.29, we have

$$\mathcal{T}_n^{\mathsf{D}} = A_{\mathcal{T}} \bigcup_{(\ell,m)} W_{(\ell,m)} = \bigcup_{(\ell,m)} A_{\mathcal{T}} W_{(\ell,m)} = \bigcup_{(\ell,m)} A_{\mathcal{T}}^{(\ell,m)} W_{(\ell,m)},$$

where the pairwise disjointness follows from the formation being a bijection. □

Theorem 3.32. *The TLG problem $(K, \mathbf{q})$ in Definition 3.8 is solved by a triangular lattice $\mathcal{T} = A_{\mathcal{T}}\mathcal{P}$ if and only if its formation $A_{\mathcal{T}}$ satisfies*

$$\forall (\ell, m) \in \mathbb{Z}_D^+ \times \mathbb{Z}_n,\ A_{\mathcal{T}}^{(\ell,m)} W_{(\ell,m)} \subset K; \tag{56a}$$

$$\exists \mathbf{p} \in \mathcal{P},\ s.t.\ A_{\mathcal{T}}\mathbf{p} = \mathbf{q}, \tag{56b}$$

where the test sets $W_{(\ell,m)}$'s form a partition of $\mathcal{P}$. Furthermore, an enumeration based on the partial D-permutations finds all solutions to the TLG problem.

Proof. This follows directly from Lemma 3.31 and Corollary 3.27. □

Theorem 3.32 is the main theoretical foundation for our TLG algorithm.

4 Algorithm

The key issue that affects the efficiency of a TLG algorithm is how to choose the equivalence classes in Definition 3.28. To this end, we give two choices of test sets in Sections 4.1 and 4.2. In Section 4.3, we propose our TLG algorithm based on the backtracking algorithm in Definition 2.8. Since we will show in Section 6 that the TLG algorithm based on test sets of type II is much more efficient than that based on test sets of type I, the reader might want to skip Section 4.1. However, by including both test tests in the exposition, we emphasize the importance of incorporating algebraic structures in the design of highly efficient algorithms.

4.1 Test sets of type I

Definition 4.1. *A function $c : \mathbb{Z}_D^+ \to \mathbb{Z}_n$ is called a* column-pick map *for a multiindex $(\ell, m) \in \mathbb{Z}_D^+ \times \mathbb{Z}_n$ if*

$$c(i) = \begin{cases} \leq m & \text{if } i < \ell; \\ m & \text{if } i = \ell; \\ < m & \text{if } i > \ell. \end{cases} \tag{57}$$

The set of all column-pick maps for a given multiindex (ℓ, m) is denoted by $C_{(\ell,m)}$.

Definition 4.2. *The* test set of type I *is a set of* D*-dimensional multiindices,*

$$W_{(\ell,m)} := \left\{ (c(1), \dots, c(\mathsf{D})) \in \mathbb{Z}_n^D : c \in C_{(\ell,m)}; \sum_{i=1}^{D} c(i) \le n \right\}. \tag{58}$$

In particular, $W_{(\ell,m)} = \emptyset$ *if* $C_{(\ell,m)} = \emptyset$.

Clearly, we have $C_{(\ell,m)} = \emptyset$ if and only if $m = 0$ and $\ell < \mathsf{D}$.

Corollary 4.3. *If* $C_{(\ell,m)}$ *is nonempty, any column-pick map* $c \in C_{(\ell,m)}$ *satisfies*

$$\forall i \in \mathbb{Z}_D^+,\ s(i, c(i)) \le s(\ell, m). \tag{59}$$

Proof. This follows from (57) and Definition 3.30. □

The following lemma justifies the name of "test set" in Definition 4.2, c.f. Definition 3.28.

Lemma 4.4. *Test sets of type I form a partition of the principal lattice* $\mathcal{P}_D^n$*; furthermore, we have*

$$(\ell, m) \neq (i, j) \ \Leftrightarrow\ W_{(\ell,m)} \cap W_{(i,j)} = \emptyset. \tag{60a}$$

Proof. Suppose $\mathbf{p} \in W_{(\ell,m)}$ for some (ℓ, m). Then $\mathbf{p} \in \mathcal{P}_{\mathsf{D}}^n$ must hold because of the condition $\sum_{i=1}^{\mathsf{D}} c(i) \le n$ in (58).

Conversely, let m be the largest coordinate of $\mathbf{p} \in \mathcal{P}_{\mathsf{D}}^n$ and ℓ be the largest dimension index of $\mathbf{p}$ such that $p_\ell = m$. Then (57) and (58) imply $\mathbf{p} \in W_{(\ell,m)}$.

Suppose for some $(i, j) \neq (\ell, m)$ we also have $\mathbf{p} \in W_{(i,j)}$. Since $\mathbf{p}$ has only one largest coordinate (that is assumed to be m), we must have $j = m$, which implies $i \neq \ell$. Because ℓ is the largest dimension index satisfying $c(\ell) = m$, we have $i < \ell$, which contradicts the third branch of (57). □

Example 4.5. *For* $\mathcal{P}_2^2$ *in (31), the test sets of type I are*

$$\begin{array}{lll} W_{(1,0)} = \emptyset, & W_{(1,1)} = \{(1,0)\}, & W_{(1,2)} = \{(2,0)\}, \\ W_{(2,0)} = \{(0,0)\}, & W_{(2,1)} = \{(0,1),(1,1)\}, & W_{(2,2)} = \{(0,2)\}. \end{array}$$

The triangular lattice in Example 2.2 is

$$\mathcal{T} = \{(0,0),(0,2),(1,0),(1,1),(1,2),(2,0)\} \tag{61}$$

and its formation is

$$A_{\mathcal{T}} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 2 & 1 \end{bmatrix}. \tag{62}$$

In light of Lemma 3.31, $\mathcal{T}$ *is partitioned into*

$$\begin{array}{l} A_{\mathcal{T}}^{(1,1)} W_{(1,1)} = \{(0,0)\},\ A_{\mathcal{T}}^{(1,2)} W_{(1,2)} = \{(2,0)\}, \\ A_{\mathcal{T}}^{(2,0)} W_{(2,0)} = \{(1,0)\},\ A_{\mathcal{T}}^{(2,1)} W_{(2,1)} = \{(1,2),(0,2)\},\ A_{\mathcal{T}}^{(2,2)} W_{(2,2)} = \{(1,1)\}. \end{array}$$

4.2 Test sets of type II

We start by examining the group action on the orbits of a subset of the G-set.

Lemma 4.6. *Let G be a group and H be a subgroup of G; let Z be a G-set and Z_0 be a subset of Z. Define*

$$W_H = \{p \in Z : O_H(p) \subset Z_0\}, \tag{63}$$

where $O_H(p)$ is the oribt of p under H, i.e. union of orbits of p under $f \in H$. Then for any $a \in G$, we have

$$aW_H = \bigcap_{b \in aH} bZ_0, \tag{64}$$

where aH is the left coset of H and each bZ_0 is a subset of Z.

Proof. Since the signature of an action of G on Z is $G \times Z \to Z$, any $a \in G$ can be considered as a permutation on Z. Hence (64) holds if and only if $W_H = \bigcap_{b \in H} bZ_0$. First, W_H is contained in bZ_0 for every $b \in H$. To show this, suppose $p \in W_H$. Since the orbit of p is contained in Z_0, we have $b^{-1}p \in Z_0$. But b is a permutation on Z; applying b on both sides we get $p \in bZ_0$. To prove the converse, suppose $p \in bZ_0$ for every $b \in H$. Applying b^{-1} on both sides we have $b^{-1}p \in Z_0$ for every $b \in H$. Since b^{-1} ranges over H, we conclude that the orbit of p under H is contained in Z_0. □

Write a sequence of subsets of $\mathbb{Z}_n$ as

$$\Lambda := (\Lambda_d)_{d=1}^{\mathsf{D}} = (\Lambda_1, \Lambda_2, \ldots, \Lambda_{\mathsf{D}}) \tag{65}$$

and denote by $H(\Lambda)$ the subset of G_n^{D} that holds fixed the positions in Λ, i.e.,

$$H(\Lambda) := \left\{A \in G_n^{\mathsf{D}} : \forall d = 1, 2, \ldots, \mathsf{D}, \forall j \in \Lambda_d,\ A(d, j) = j\right\}. \tag{66}$$

It is straightforward to verify that $H(\Lambda)$ is a subgroup of G_n^{D}. Two D-permutations $A, B \in G_n^{\mathsf{D}}$ are in the same coset of $H(\Lambda)$ if and only if their actions agree on each Λ_d in Λ,

$$\forall d = 1, \cdots, \mathsf{D},\ \forall j \in \Lambda_d,\ A(d, j) = B(d, j).$$

What is the orbit of a point $p \in \mathbb{Z}_n^{\mathsf{D}}$ under $H(\Lambda)$? For a fixed dimension d, if $p_d \in \Lambda_d$, all permutations in $H(\Lambda)$ will hold p_d fixed; otherwise $p_d \notin \Lambda_d$, then p_d will be mapped to one of $\mathbb{Z}_n \setminus \Lambda_d$. Since each dimension of a D-permutation can be assigned independently, the orbit is the Cartesian product

$$O_{H,\Lambda}(p) := \prod_{d=1}^{\mathsf{D}} W^+(p_d, \Lambda_d), \quad W^+(c, S) := \begin{cases} \{c\} & \text{if } c \in S; \\ \mathbb{Z}_n \setminus S & \text{if } c \notin S. \end{cases} \tag{67}$$

Lemma 4.6 and the above observations yield the following central result.

Theorem 4.7. *For a sequence Λ of subsets of $\mathbb{Z}_n$, define*

$$W(\Lambda) := \left\{ p \in \mathbb{Z}_n^D : O_{H,\Lambda}(p) \subset \mathcal{P} \right\}, \tag{68}$$

where $O_{H,\Lambda}(p)$ is defined in (67) and $\mathcal{P}$ is the corresponding principal lattice. Then for any D-permutation $A \in G_n^D$, we have

$$AW(\Lambda) = \bigcap_{B \in \mathcal{B}_A} B\mathcal{P}, \tag{69}$$

where $\mathcal{B}_A$ is the set of D-permutations whose actions agree with that of A on each Λ_d in Λ.

By (68), we have $W(\Lambda) = \emptyset$ if $\Lambda_1 = \cdots = \Lambda_D = \emptyset$ and $W(\Lambda) = \mathcal{P}$ if $\Lambda_1 = \cdots = \Lambda_D = \mathbb{Z}_n$. We also have

$$\forall d = 1, \cdots, \mathsf{D},\ \Lambda'_d \subset \Lambda_d \quad \Rightarrow \quad W(\Lambda') \subset W(\Lambda)$$

because then $H(\Lambda)$ is a subgroup of $H(\Lambda')$, c.f. (66) and (68).

Denote by $A|_\Lambda$ the partial permutation that results from restricting A onto the one-dimensional subsets in $\Lambda = (\Lambda_d)_{d=1}^{\mathsf{D}}$. If we use the linear ordering in Definition 3.30 to organize all formations into a spanning tree, the collection $\mathcal{B}_A$ consists of the leaf nodes of all subtrees rooted at $A|_\Lambda$. Theorem 4.7 guarantees that the set $AW(\Lambda)$ will reside in every transformed lattice $B\mathcal{P}$ ($B \in \mathcal{B}_A$) no matter which path one follows from the node $A|_\Lambda$. Therefore, the whole subtree rooted at $A|_\Lambda$ can be eliminated if the test on $AW(\Lambda)$ against feasible nodes in K fails. Theorem 4.7 also suggests that the choice of $W(\Lambda)$ is optimal, because it is the largest possible subset of $\mathbb{Z}_n^{\mathsf{D}}$ subject to $AW(\Lambda) \subset \bigcap B\mathcal{P}$.

The above discussions explain the ideas behind the following definition.

Definition 4.8. *A* test set of type II *is a subset of $\mathbb{Z}_n^D$,*

$$W_{(\ell,m)} := W(\Lambda^k) \setminus W(\Lambda^{k-1}), \tag{70}$$

where $W(\Lambda^k)$ is defined in (68), $k = s(\ell, m) = 1, 2, \ldots, \mathsf{D}(n+1)$ as in Definition 3.30, $\Lambda^0 := (\emptyset)_1^D$, $\Lambda^k := (\Lambda_d^k)_{d=1}^D$, and

$$\Lambda_d^k = \begin{cases} \emptyset & \text{if } d > \ell \text{ and } m = 0; \\ \{0, 1, \ldots, m-1\} & \text{else if } d > \ell; \\ \{0, 1, \ldots, m\} & \text{otherwise.} \end{cases} \tag{71}$$

By (71), Λ^{k+1} has one more element than Λ^k; also, we have a filtration

$$\Lambda^0 \subset \Lambda^1 \subset \cdots \subset \Lambda^{\mathsf{D}(n+1)} := (\mathbb{Z}_n)_1^{\mathsf{D}}$$

that corresponds to the enumeration order of the D-permutations. Then by (68), test sets of type II indeed form a partition of $\mathcal{P}$, c.f. Definition 3.28.

Example 4.9. *For $\mathcal{P}_2^2$ in (31), the test sets of type II are*

$$\begin{array}{lll} W_{(1,0)} = \{(0,0),(0,1),(0,2)\}, & W_{(1,1)} = \emptyset, & W_{(1,2)} = \emptyset, \\ W_{(2,0)} = \{(1,0),(2,0)\}, & W_{(2,1)} = \{(1,1)\}, & W_{(2,2)} = \emptyset. \end{array}$$

As suggested by a comparison of Examples 4.5 and 4.9, test sets of type II indeed have an advantage over those of type I: test sets to be checked early contain a larger number of points, thus a failure of an early test set eliminates more subtrees in the solution space.

4.3 Backtracking for TLG

To solve the TLG problem, we specialize the backtracking algorithm in Definition 2.8 by giving specific definitions to the itemized subroutines in `typewriter` font. We begin by defining an ordering that is useful for the subroutines `first` and `next`.

Definition 4.10. *For a TLG problem $(K, \mathbf{q})$, the (compactness-first)* test ordering *"$<$" of a subset $J_i \subseteq \mathbb{Z}_n$ along the ith dimension is a total ordering on J_i determined first by the distance to $\mathbf{q}$ along the ith dimension and then by breaking any tie with the cardinality of the slices. More precisely, for any distinct $j, k \in J_i$, we have*

$$j < k \;\Leftrightarrow\; \big(|j-q_i| > |k-q_i|\big) \;\vee\; \big(|j-q_i| = |k-q_i| \wedge \#L_{i,j}(K) < \#L_{i,k}(K)\big), \tag{72}$$

where q_i is the ith coordinate of $\mathbf{q}$. In particular, the element in J_i that is greater and less than all other elements in J_i is denoted by $\max J_i$ *and* $\min J_i$*, respectively.*

Note that the test ordering is just one particular choice of traversing the spanning tree. There might as well exist many other ways to traverse the solution space. For example, the feasibility-first test ordering is a total ordering on J_i determined first by the cardinality of the slices along the ith dimension and then by breaking any tie with the distance to $\mathbf{q}$, i.e.,

$$j < k \;\Leftrightarrow\; \big(\#L_{i,j}(K) < \#L_{i,k}(K)\big) \;\vee\; \big(\#L_{i,j}(K) = \#L_{i,k}(K) \;\wedge\; |j-q_i| > |k-q_i|\big). \tag{73}$$

The set of *all* solutions of a given TLG problem does not depend on the order of traverse; after all, one never knows whether the current set of solutions is the set of all solutions until she has exhausted all possibilities.

We might be happy with finding just one solution of the TLG problem, in which case we can save some time by stopping the traverse with the first solution.

Definition 4.11 (A backtracking algorithm for TLG)**.** *The following recursive algorithm finds all solutions to the TLG problem in Definition 3.8.*

> **Input:** *the degree* $n \in \mathbb{N}^+$*, the dimensionality* $\mathsf{D} \in \mathbb{N}^+$*, the search domain* $K \subseteq \mathbb{Z}_n^D$*, and the starting point* $\mathbf{q} \in K$
> **Output:** *a set* $\mathcal{U}$ *of* D*-permutations*
> **Postconditions:** $\{A\mathcal{P}_D^n : A \in \mathcal{U}\}$ *is the set of all solutions of the TLG problem*
>
> $\mathcal{U} \leftarrow$ *an empty set of* D*-permutations*
> $A^{(t)} \leftarrow$ `root`(n, D)
> **BackTrack** $\big((K, \mathbf{q}), A^{(t)}, \mathcal{U}\big)$

where the procedures in Definition 2.8 are as follows.

- `root`(n, D)*: set* $A^{(0)}$ *to a* D*-by-*$(n+1)$ *matrix of constant* -1 *following Notation 2; return* $A^{(0)}$*.*
- `accept`$((K, \mathbf{q}), A^{(t)})$*: return false if* $t < \mathsf{D}(n+1)$*; otherwise return true if and only if*
$$A^{(t)} W_{(D,n)} \subset K \text{ and } \mathbf{q} \in A^{(t)} \mathcal{P}_D^n.$$
- `stopAfterAccept`$((K, \mathbf{q}), A^{(t)})$*: return true.*
- `reject`$((K, \mathbf{q}), A^{(t)})$*: if* $t = \mathsf{D}(n+1)$*, return the negation of* `accept`$((K, \mathbf{q}), A^{(t)})$*; otherwise return the value of the logical statement*
$$\exists \mathbf{p} \in W_{(\ell,m)} \text{ s.t. } A^{(t)}\mathbf{p} \notin K,$$
where $(\ell, m) = s^{-1}(t)$*.*
- `first`$((K, \mathbf{q}), A^{(t)})$*: let* $(\ell, m) = s^{-1}(t+1)$ *and initialize*
$$B^{(t+1)} \leftarrow A^{(t)},$$
$$J_\ell := \mathbb{Z}_n \setminus \left\{ B^{(t+1)}(\ell, j) : j = 0, 1, \ldots, m-1 \right\}. \tag{74}$$
Set $B^{(t+1)}(\ell, m) \leftarrow \max J_\ell$ *as in Definition 4.10 and return* $B^{(t+1)}$*.*
- `next`$((K, \mathbf{q}), A^{(t)}, B^{(t+1)})$*: let* $(\ell, m) = s^{-1}(t+1)$ *and compute* J_ℓ *by (74). Return null if* $B^{(t+1)}(\ell, m)$ *equals* $\min J_\ell$ *as in Definition 4.10. Otherwise initialize* $C^{(t+1)} \leftarrow B^{(t+1)}$*, set* $C^{(t+1)}(\ell, m)$ *to be the element in* J_ℓ *that is* immediately *smaller than* $B^{(t+1)}(\ell, m)$*, and return* $C^{(t+1)}$*.*

The above specifications of subroutines entail some explanation. First, a partial D-permutation is formally not a D-permutation until the last stage $t = \mathsf{D}(n+1)$; this is the reason for the conditions $t < \mathsf{D}(n+1)$ and $t = \mathsf{D}(n+1)$ in the subroutines `accept` and `reject`, respectively. However, in the case of test sets of type II, the partial D-permutation has already been determined at the stage $t = \mathsf{D}n$; see Example 4.9. Second, the subroutine `stopAfterAccept`

always returns true because a D-permutation, once determined, can never be augmented to another one. Third, the subroutine `first` returns the first child node of the parent node $A^{(t)}$, hence in (74) we should remove from $\mathbb{Z}_n$ the numbers that are already in the same row of $A^{(t)}$. Lastly, the subroutine `next` returns the child node of the parent node $A^{(t)}$ that is immediately after the child node $B^{(t+1)}$. Hence $B^{(t+1)} = \min \mathcal{J}_\ell$ means that all child nodes of $A^{(t)}$ have already been checked.

5 The PLG-FD method

Based on the TLG algorithm in Section 4, we propose a new FD method as follows.

Definition 5.1 (The PLG-FD method)**.** *For an elliptic equation*

$$\mathcal{L}u(\mathbf{x}) = 0 \text{ on } \Omega \tag{75}$$

of a scalar function $u : \mathbb{R}^D \to \mathbb{R}$ with appropriate boundary conditions on $\partial\Omega$, the PLG-FD method *produces discrete solutions of (75) by four steps as follows.*

(a) Embed $\Omega \subset \mathbb{R}^{\mathsf{D}}$ in a rectangular Cartesian grid. *For ease of exposition the grid is assumed to have a uniform grid size h for all dimensions. Then each cell can be indexed by a multiindex $\mathbf{i} \in \mathbb{Z}^D$ with its center at*

$$\mathbf{x}_\mathbf{i} = h\mathbf{i} + \frac{1}{2}h\mathbb{1},$$

where $\mathbb{1}$ is the multiindex whose components are all 1's.

(b) Classify the cell centers.

- *A cell center $\mathbf{x}_\mathbf{i}$ is called an* exterior node *if $\mathbf{x}_\mathbf{i} \notin \Omega$ and its distance to $\partial\Omega$ is greater than ηh, where η is a user-specified small positive number.*
- *A cell center $\mathbf{x}_\mathbf{j}$ is called a* boundary node *if $\mathbf{x}_\mathbf{j}$ is not an exterior node but a face neighbor of an exterior node $\mathbf{x}_\mathbf{i}$, i.e., $\|\mathbf{j} - \mathbf{i}\|_1 = 1$. Each boundary node is also associated with a* boundary point*, i.e., the point on $\partial\Omega$ that is closest to the boundary node.*
- *All other cell centers are called* interior node*s.*

Boundary nodes and interior nodes are collectively called FD node*s, where discrete solutions of the PDE are sought. See Figure 3 for an illustration.*

(c) Discretize the PDE for each FD node.

- *If the FD node* $\mathbf{x_i}$ *is a* regular FD node*, i.e. an interior node away from the irregular boundary such that all points in its standard one-dimensional FD stencils are FD nodes, we use these standard stencils and their tensor products to discretize* $\mathcal{L}$ *at* $\mathbf{x_i}$*. For example, for fourth-order accuracy in two dimensions, the Laplacian is discretized with the standard 9-point stencil and the cross derivative term* $\frac{\partial^2 u}{\partial x \partial y}$ *with a 5-by-5 square box.*
- *Otherwise* $\mathbf{x_i}$ *is called* an irregular FD node*. We generate a poised lattice for* $\mathbf{x_i}$ *from nearby FD nodes.*
  - *If* $\mathbf{x_i}$ *is an interior node, we generate a poised lattice that consists of nearby FD nodes (as an answer to the problem in Definition 1.2), denoted by* $\mathbf{S(i)}$*. Approximating* u *on each FD node using a multivariate polynomial* $P_\mathbf{i} = \sum_{k=1}^{N} c_{\mathbf{i},k}\phi_k(\mathbf{x})$*, we get an equation for each node* $\mathbf{j} \in \mathbf{S(i)}$:

$$u(\mathbf{x_j}) \approx P_\mathbf{i}(\mathbf{x_j}) = \sum_{k=1}^{N} b_{\mathbf{j},k} c_{\mathbf{i},k}, \tag{76}$$

    where $b_{\mathbf{j},k}$ *depends on the basis function* ϕ_k *and the position of the sampling sites for node* $\mathbf{j}$*. Since the cardinality of the stencil* $\mathbf{S(i)}$ *equals* $\dim \Pi_n^D$*, (76) leads to a linear system*

$$\mathbf{B_i c_i} = \mathbf{U_{S(i)}}, \tag{77}$$

    where the $\mathbf{j}$*th row of the sample matrix* $\mathbf{B_i}$ *are the vector* $\mathbf{b_j}$ *that consists of* $b_{\mathbf{j},k}$*'s, the unknown vector* $\mathbf{c_i}$ *consists of* $c_{\mathbf{i},k}$*'s, and* $\mathbf{U_{S(i)}}$ *the value* u *on the nodes in the stencil* $\mathbf{S(i)}$*. Solving (77) yields* $\mathbf{c_i}$*, and then plug the fitted multivariate polynomial into (75) to obtain the discretized PDE at* $\mathbf{x_i}$*.*
  - *Otherwise* $\mathbf{x_i}$ *is a boundary node. The procedures to obtain the discretized PDE are almost the same as those of the previous case except that we incorporate the boundary condition at the boundary point* $b_\mathbf{i}$ *by adding an extra row in the sample matrix (2) and solve the resulting least squares problem. Note that for a Neumann boundary condition, the extra row in the sample matrix reads*

$$\left[\frac{\partial \phi_1}{\partial \mathbf{n}}(b_\mathbf{i}),\ \cdots,\ \frac{\partial \phi_N}{\partial \mathbf{n}}(b_\mathbf{i}) \right],$$

    where $\mathbf{n}$ *is the outward normal of* $\partial\Omega$*, and* ϕ_i*'s the monomial basis of* Π_n^D*.*

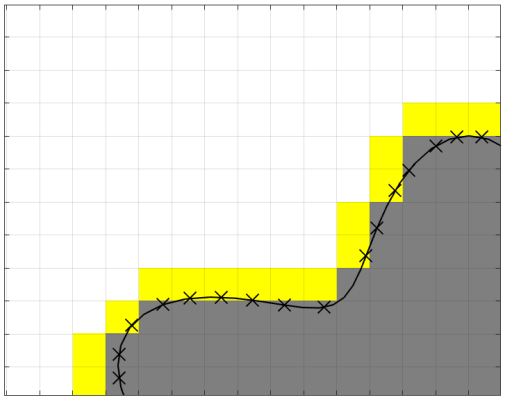
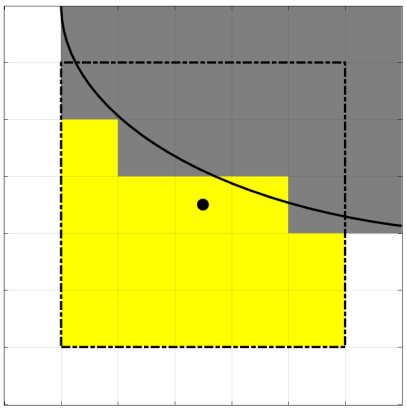
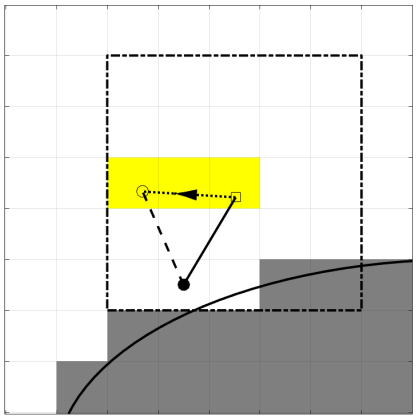


Figure 3: Classify the cell centers in the PLG-FD method. The curve represents part of the domain boundary; a gray square represents an exterior node, a white square an interior node, a yellow square a boundary node, and a cross mark a boundary point.



(a) Discretizing Δu. The first pivot set only contains the starting point $\mathbf{q}$ and the resulting set K contains all yellow cells.



(b) Discretizing $\mathbf{v} \cdot \nabla u$. The first and the last pivot sets are $\{\mathbf{p}_1\}$ and $\{\mathbf{p}_2\}$, where $\mathbf{p}_1$ and $\mathbf{p}_2$ are defined in (80) and represented by the hollow square and the hollow dot, respectively. The set K resulting from $\mathbf{p}_1$ contains the yellow cells and white cells inside the dot-dashed box.

Figure 4: Our strategy (KQN) to determine the set K of feasible nodes from an even n and an irregular FD node $\mathbf{q}$ for TLG discretization in the geometric FD method. A solid dot represents the irregular FD node, to which the starting point $\mathbf{q}$ is always assigned. Gray squares represent exterior nodes and the curve the domain boundary $\partial\Omega$. A box of dot-dashed line represents a shifted cube centered at the first pivot. Yellow squares represent pivots in the sequence of pivot sets.

(d) Solve the system of discrete equations to produce the numerical solution.

For the time-dependent problems, such as the heat equations and the advection-diffusion equations, we use the method of lines (MOL). This technique for solving PDEs involves: (a) discretizing the spatial derivatives while leaving the time variable continuous; (b) solving the resulting ODEs with a numerical method designed for IVPs.

The above PLG-FD method is appealing in a number of aspects.

First, thanks to the ease of taking derivatives of polynomials, algorithmic steps of PLG-FD are decoupled from the specific form and boundary conditions of the PDE. This is different from other methods such as that in [12] where the discretization of spatial operators is coupled to the PDE and boundary/interface conditions.

Second, the steps in Definition 5.1 directly apply to a wide array of PDEs. For elliptic equations, adding a cross derivative term such as $\frac{\partial^2 u}{\partial x \partial y}$ incurs no changes on algorithmic steps. Coupled to a time integrator, the PLG-FD method is also an efficient solver for parabolic equations and other time-dependent problems.

Third, the PLG-FD method achieves high-order accuracy while retaining the simplicity of traditional FD methods. For second- and lower-order methods (such as that in [14]), the problem of choosing interpolation sites in fitting a polynomial is trivial for $n = 1, 2$, but not so for higher values of n. Some authors resort to least squares by adding a large number of interpolation sites in polynomial fitting. However, least squares do not guarantee the nonsingularity of the linear system and a large number of redundant interpolation sites might deteriorate the efficiency. In contrast, least squares in the PLG-FD method start with a poised lattice and the balance between conditioning and efficiency is completely under control. See the last paragraph in Section 6.1 for more discussions.

Lastly, the TLG algorithm is dimensionality-agnostic and furnishes certain degree of flexibility in catering for the physics of the operators to be discretized.

5.1 How to choose the feasible set in PLG-FD?

In this subsection we suggest a choice of the feasible set according to the physics of the problem at hand. For example, to discretize the Laplacian Δu, the generated lattice should be centered at the starting point as much as possible. In comparison, for the advection operator $\mathbf{v} \cdot \nabla u$, we typically have a lopsided lattice to capture the physics that most information comes from the upwind direction $-\mathbf{v}$.

(CFS-1) From n and $\mathbf{q}$, we determine a *sequence of pivot sets* $O_j \subset \mathbb{R}^{\mathsf{D}}$.

Denote the union of cell corners and cell centers by

$$V := (h\mathbb{Z})^{\mathsf{D}} \cup \left(h\mathbb{Z} + \frac{h}{2}\mathbb{1} \right)^{\mathsf{D}}. \tag{78}$$

For the Laplacian Δu, we choose

$$O_j^{\mathrm{Lap}} = \left\{ \mathbf{x} \in V : \frac{1}{h}\|\mathbf{x} - \mathbf{q}\|_\infty = j - 1 + \frac{1}{2}(n \bmod 2) \right\} \tag{79}$$

with $j = 1, 2, \ldots, \alpha_n$ and $\alpha_n := \lfloor \frac{n}{2} \rfloor$. For the advection $(\mathbf{v} \cdot \nabla)u$, we use (79) if $\|\mathbf{v}\|_2$ is sufficiently small; otherwise we define

$$\mathbf{p}_1 := \mathbf{q} - \alpha_n h \frac{\mathbf{v}}{\|\mathbf{v}\|_2}, \qquad \mathbf{p}_2 := \mathbf{q} - \alpha_n h \mathbf{n}_b, \tag{80}$$

where $\mathbf{n}_b$ is the unit outward normal of $\mathbf{q}_b \in \partial\Omega$ and $\mathbf{q}_b$ is the closest point to $\mathbf{q}$ on $\partial\Omega$. As a particle travels from $\mathbf{p}_1$ to $\mathbf{p}_2$, it intersects a sequence of cells whose centers are FD nodes $\mathbf{x}_j^{\mathrm{adv}}$. Then we choose

$$O_j^{\mathrm{Adv}} = \left\{ \mathbf{x} \in V : \frac{1}{h}\|\mathbf{x} - \mathbf{x}_j^{\mathrm{adv}}\|_\infty = \frac{1}{2}(n \bmod 2) \right\}. \tag{81}$$

(CFS-2) For each pivot $\mathbf{x} \in O_j$, we shift $\mathbb{Z}_n^{\mathsf{D}}$ to center it at $\mathbf{x}$, identify K with the set of FD nodes in the shifted cube, and solve the resulting TLG problem to obtain at most one poised lattice.

(CFS-3) If the previous step yields multiple poised lattices, we select the lattice $\{\mathbf{x_j}\}$ of which $\sum_{\mathbf{j}} \|\mathbf{x_j} - \mathbf{q}\|_2^2$ is minimized.

(CFS-4) Otherwise no solution exists for any pivot in O_j. We increase j by 1 and repeat (CFS-1,2,3) until we have a solution or until, for the given n, no solution exists for any pivot set in the sequence.

In Figure 4 we illustrate the above strategy in the case of n being an even number. Numerical experiments show that $j = 1$ in (CFS-1) yields a poised lattice in most cases.

6 Tests

In Section 6.1, we test the TLG algorithm by generating poised lattices near irregular boundaries. The variation of condition number of the sample matrix with respect to the number of points in least square stencils is also studied for representative scenarios. In Section 6.2, we test the TLG discretization

of a spatial operator to show fourth- and sixth-order convergence of truncation errors. In Section 6.3, we demonstrate the effectiveness of PLG-FD by solving an elliptic equation with a cross-derivative term. In Section 6.4, we show that the fourth-order PLG-FD method is more accurate than a previous second-order EB method in solving Poisson's equation. In Section 6.5, we test Poisson's equation in three dimensions. In Section 6.6 and 6.7, we consider the time-dependent problems, demonstrating the fourth-order accuracy of our method for both the heat equation and the convection-diffusion equation.

6.1 TLG near irregular boundaries

By Definition 3.8, a TLG problem is uniquely specified by the feasible set K and the starting node $\mathbf{q}$. Our design of test cases has been centered around representative scenarios in the context of FD methods.

First, although the dimensionality D can be an arbitrary positive integer, we only test the cases of $\mathsf{D} = 2$ and $\mathsf{D} = 3$.

Second, the starting point $\mathbf{q}$ is an irregular FD node $\mathbf{x_i}$ that is either a boundary node or an interior node close to the boundary; see Definition 5.1 (c).

Third, in the asymptotic case of local geometry being well resolved by the rectangular grid, the set K is well described either by one hyperplane or by multiple hyperplanes. The former corresponds to a smooth boundary surface while the latter a boundary surface with discontinuous normal vectors. In both cases, however, K must lie entirely at one side of the boundary surface. In light of these observations, for $\mathsf{D} = 2$ we select the irregular boundary to be either a rotated box or an ellipse

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1. \tag{82}$$

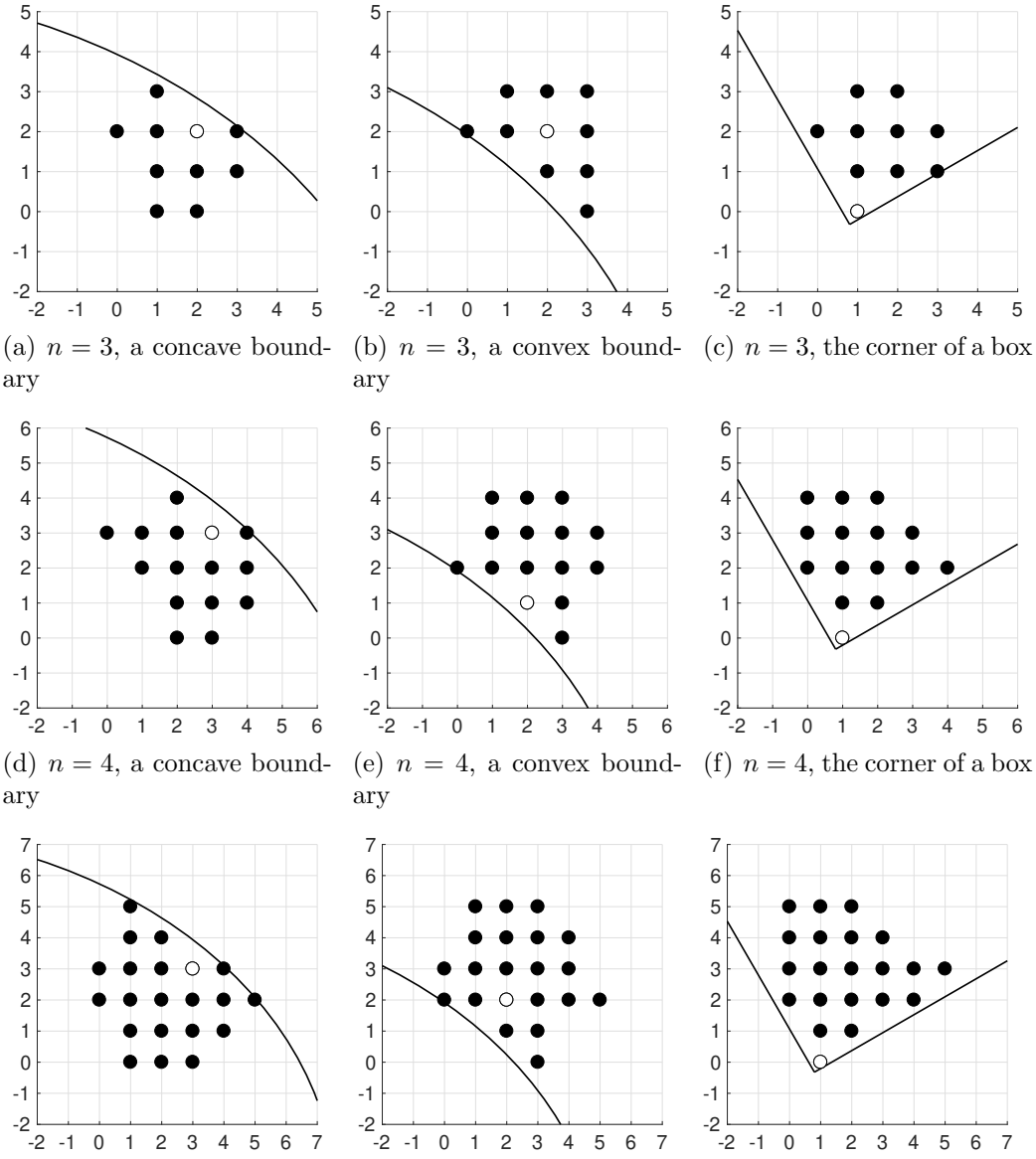
For $\mathsf{D} = 3$, the irregular boundary is either the surface of a rotated cube or an ellipsoid

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} + \frac{(z - z_0)^2}{c^2} = 1. \tag{83}$$

The above characterizations of test cases are orthogonal, which helps the enumeration of representative TLG problems. We invoke the TLG algorithm in Definition 4.11 on all these TLG problems, stop on finding the first poised lattice, and list results of some of them in Figures 5 and 6.

By Definition 3.1, all lattices shown in Figures 5 and 6 are indeed triangular lattices because each lattice contains $\binom{n+\mathsf{D}}{n}$ nodes and projecting these nodes to each axis yields exactly $n + 1$ distinct coordinates.

We further test how the number of redundant points in a least-square stencil affects the condition number of the sample matrix M in (2) for the

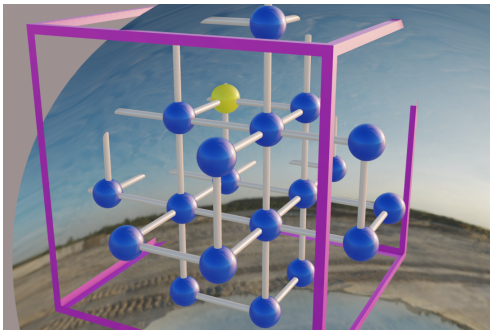


(a) $n = 3$, a concave boundary (b) $n = 3$, a convex boundary (c) $n = 3$, the corner of a box

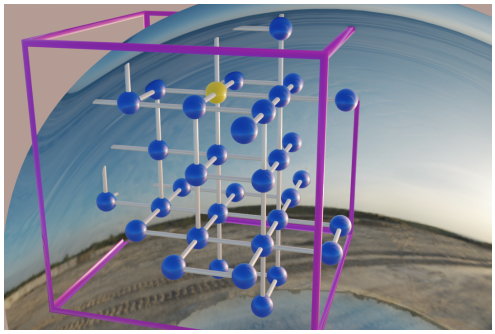
(d) $n = 4$, a concave boundary (e) $n = 4$, a convex boundary (f) $n = 4$, the corner of a box

(g) $n = 5$, a concave boundary (h) $n = 5$, a convex boundary (i) $n = 5$, the corner of a box

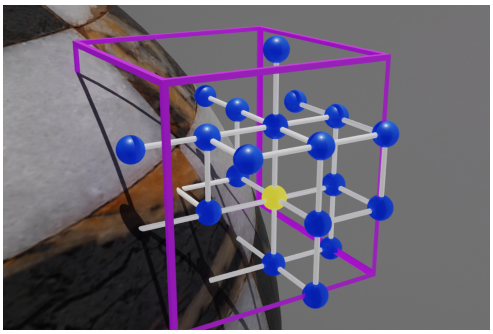
Figure 5: Poised lattices generated by the TLG algorithm in Definition 4.11 with the compactness-first test ordering (72). In all cases, the starting point $\mathbf{q}$ is represented by the hollow circle and K is the set of FD nodes in the shifted cube $\mathbb{Z}_n^2$ that contains all hollow and solid dots. $\mathbf{q}$ is an interior node in subplots (b,h) and is a boundary node in all other cases. All smooth boundaries are arcs of ellipses in (82) with $(a, b) = (16, 12)$. We have $(x_0, y_0) = (-8.40, -6.28)$ in subplot (a), $(x_0, y_0) = (-8.40, -4.48)$ in subplots (d,g), and $(x_0, y_0) = (-10.88, -6.88)$ in subplots (b,e,h). In subplots (c,f,i), the lower corner of each box is at $(0.80, -0.32)$ with the right side angled at $\frac{\pi}{6}$.



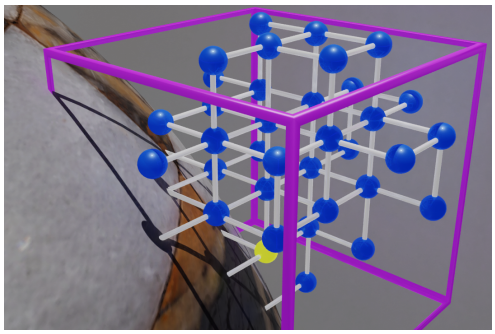
(a) $n = 3$, $x_0 = y_0 = z_0 = 0$, $a = b = c = 3.6$



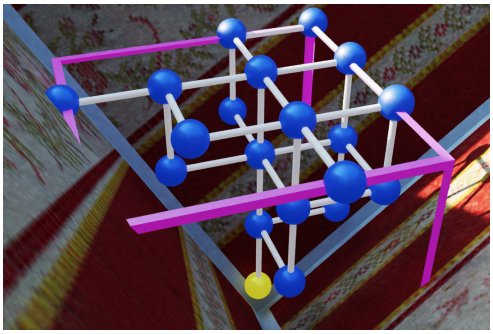
(b) $n = 4$, $x_0 = y_0 = z_0 = 0$, $a = b = c = 4.8$



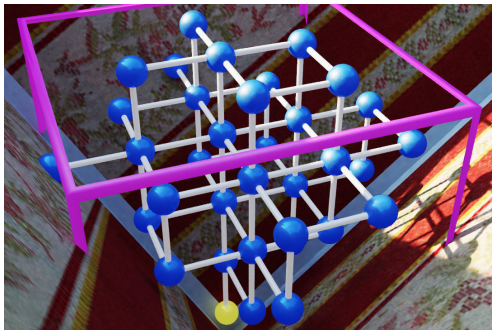
(c) $n = 3$, $(x_0, y_0, z_0) = (12, 9, -3)$, $(a, b, c) = (15, 12, 9)$



(d) $n = 4$, $(x_0, y_0, z_0) = (16, 12, -4)$, $(a, b, c) = (20, 16, 12)$

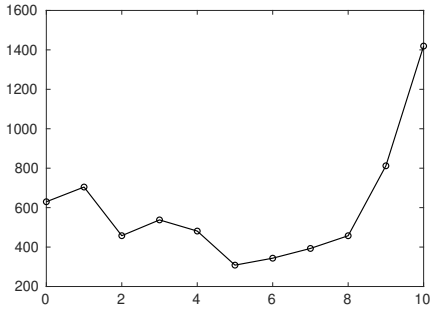


(e) $n = 3$, the box corner is at $(1.0, 1.0, -0.5)$

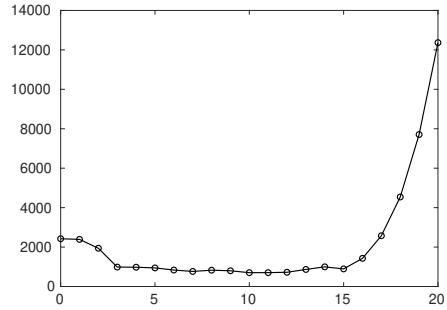


(f) $n = 4$, the box corner is at $(2.0, 2.0, -0.5)$

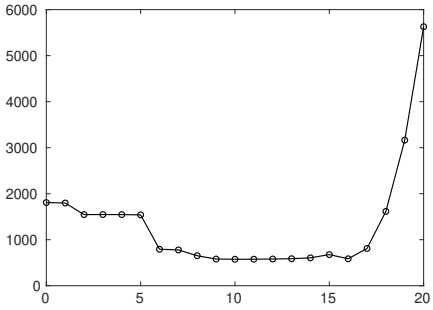
Figure 6: Poised lattices generated by the TLG algorithm in Definition 4.11 with the compactness-first test ordering (72). In all cases, the starting point $\mathbf{q}$ is represented by the yellow ball and the cube $\mathbb{Z}^3_n$ is shifted to coincide with the blue frame. $\mathbf{q}$ is an interior node in subplot (c) and is a boundary node in all other cases. All smooth boundaries are patches of the ellipsoid in (83). In subplots (e,f), the box is obtained by rotating the unit cube with Euler angles $\frac{\pi}{4}$. Note that any slice of the lattice along any dimension is a two-dimensional triangular lattice. For $\mathsf{D} = 3$ and $n = 3$, the result of the TLG algorithm with the feasibility-first test ordering (73) is shown in Figure 3(c).



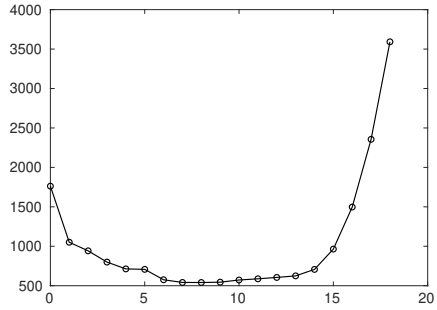
(a) for Figure 5(c) with $\mathsf{D} = 2$ and $n = 3$



(b) for Figure 5(e) with $\mathsf{D} = 2$ and $n = 4$



(c) for Figure 6(b) with $\mathsf{D} = 3$ and $n = 4$



(d) for Figure 6(e) with $\mathsf{D} = 3$ and $n = 3$

Figure 7: Condition numbers of the sample matrix (2) for least square stencils based on poised lattices. Each subplot is generated by adding redundant points, one at a time, into a poised lattice in Figure 5 or Figure 6 and computing the resulting condition number. The abscissa is the number of redundant points and the ordinate is the condition number of the sample matrix. Redundant points with smaller Manhattan distance to the starting point are added before those with larger Manhattan distances.

least-square problem. Starting with a poised lattice in Figure 5 or 6, we add redunant points one at a time into the stencil, calculate the corresponding condition number of M, and present the results in Figure 7, in which we observe several prominent features. First, the condition number is reduced as we increase the number of redundant points; there exists a (typically flat) minimum of condition numbers. Second, further adding redundant points into the stencil might lead to an increase of the condition number, thus a large number of redundant points in a least square stencil might be detrimental to both efficiency and conditioning. Third, since a poised lattice has no redundant points, it admits a fine control of the least square stencil to strike a good balance between efficiency and conditioning.

Table 1: CPU Time of TLG discretization of $\nabla \cdot (\mathbf{uu})$ on an Intel Xeon CPU E5-2698 v3 at 2.30GHz.

			number of	CPU time in seconds		
D	n	h	TLG calls	test sets of type I	test sets of type II	ratio
2	4	$\frac{1}{64}$	192	5.54e−3	6.49e−4	8.54e+0
2	6	$\frac{1}{64}$	288	6.83e+0	1.66e−2	4.11e+2
2	8	$\frac{1}{64}$	384	2.99e+4	1.61e+1	1.86e+3
2	8	$\frac{1}{128}$	768	4.62e+4	3.20e+1	1.44e+3
3	4	$\frac{1}{32}$	1696	2.03e+0	1.33e−2	1.53e+2
3	6	$\frac{1}{32}$	2549	4.19e+4	1.81e+1	2.31e+3

6.2 The PLG discretization of continuous operators

We start with an elaboration of the PLG discretization, i.e., Step(c) of the PLG-FD method in Definition 5.1. For each irregular FD node $\mathbf{x_j}$, we set $\mathbf{q} = \mathbf{x_j}$, determine K from $\mathbf{q}$ and n using the (CFS) strategy in Section 5.1, invoke the TLG algorithm to solve the TLG problem, identify the generated poised lattice $\mathcal{T}_\mathbf{j} = \{\mathbf{x_i}\}$ with the set of sites in the LIP in Definition 1.1, and solve the LIP to obtain a multivariate polynomial $p(\mathbf{x})$ whose coefficients are functions of point values $u(\mathbf{x_i})$'s. Then we take derivatives of $p(\mathbf{x})$ as dictated by $\mathcal{L}u$ to obtain a discrete form $Lu(\{\mathbf{x_i}\}) \approx \mathcal{L}u(\mathbf{x_j})$. Note that L is another polynomial of the $u(\mathbf{x_i})$'s.

For a spatial operator $\mathcal{L}$ on a scalar function u, the *truncation error for the TLG discretization of* $\mathcal{L}u$ at an irregular FD node $\mathbf{x_j}$ is

$$E_T(\mathcal{L}u, \mathbf{x_j}) := |Lu(\{\mathbf{x_i}\}) - \mathcal{L}u(\mathbf{x_j})|\,, \tag{84}$$

where $\{\mathbf{x_i}\}$ is the poised lattice of $\mathbf{x_j}$ generated by the TLG algorithm and L is the TLG discretization of $\mathcal{L}$ as described in the previous paragraph.

In this test, the irregular boundaries in two and three dimensions are respectively the ellipse and the ellipsoid in (82) and (83). The scalar function u is set to be each component of a velocity field $\mathbf{u}$, which, in two and three dimensions, is respectively given by

$$\mathbf{u}(x,y) = \begin{pmatrix} \sin^2(\pi x)\sin(2\pi y) \\ -\sin(2\pi x)\sin^2(\pi y) \end{pmatrix}, \tag{85}$$

$$\mathbf{u}(x,y,z) = \frac{1}{2}\begin{pmatrix} \sin^2(\pi x)\sin(2\pi y)\sin(2\pi z) \\ \sin(2\pi x)\sin^2(\pi y)\sin(2\pi z) \\ -2\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z) \end{pmatrix}. \tag{86}$$

In Sections 4.1 and 4.2, two types of test sets are given for use with the TLG algorithm. In this test, we first compare CPU time of these two different test sets and list the results in Table 1. Test sets of type II are clearly much

Table 2: Truncation errors for the TLG discretization of $\nabla \cdot (\mathbf{uu})$; norms are taken over all components of $\mathbf{u}$. The two-dimensional domain is the unit square with an ellipse removed, prescribed by (82) with $(x_0, y_0) = (\frac{1}{2}, \frac{1}{2})$ and $(a, b) = (\frac{1}{4}, \frac{1}{8})$. The three-dimensional domain is the unit cube with an ellipsoid removed, prescribed by (83) with $(x_0, y_0, z_0) = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ and $(a, b, c) = (\frac{1}{4}, \frac{1}{8}, \frac{1}{4})$.

$\mathsf{D} = 2$, $n = 4$							
	$h = \frac{1}{32}$	rate	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$	rate	$h = \frac{1}{256}$
L^∞	4.40e-03	3.51	3.87e-04	3.69	3.01e-05	3.84	2.10e-06
L^1	5.41e-04	4.08	3.20e-05	4.00	1.99e-06	4.01	1.24e-07
L^2	5.78e-04	4.18	3.19e-05	4.00	1.99e-06	4.05	1.21e-07
$\mathsf{D} = 2$, $n = 6$							
	$h = \frac{1}{32}$	rate	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$	rate	$h = \frac{1}{256}$
L^∞	2.99e-04	6.81	2.66e-06	4.15	1.49e-07	5.30	3.80e-09
L^1	2.08e-05	6.22	2.80e-07	6.02	4.33e-09	6.02	6.67e-11
L^2	2.78e-05	6.59	2.87e-07	5.88	4.89e-09	6.05	7.38e-11
$\mathsf{D} = 3$, $n = 4$							
	$h = \frac{1}{32}$	rate	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$		
L^∞	6.74e-03	3.04	8.21e-04	4.50	3.63e-05		
L^1	2.88e-04	4.01	1.79e-05	4.01	1.11e-06		
L^2	2.85e-04	4.09	1.67e-05	4.07	9.94e-07		
$\mathsf{D} = 3$, $n = 6$							
	$h = \frac{1}{32}$	rate	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$		
L^∞	4.50e-04	4.78	1.64e-05	6.38	1.96e-07		
L^1	9.72e-06	6.01	1.50e-07	6.03	2.30e-09		
L^2	1.29e-05	6.11	1.87e-07	6.30	2.38e-09		

more efficient than those of type I. Furthermore, the ratio of improvement increases as n grows, but it appears to be insensitive to the grid size, c.f. the third and fourth rows in Table 1. These observations confirm our conjecture that the backtracking efficiency can be greatly enhanced by enforcing in the spanning trees the algebraic structure of orbits of equivalence classes of the principal lattice.

When each component of $\mathbf{u}$ is approximated by a complete multivariate polynomial of degree no more than n, the leading error term of $\mathbf{uu}$ is $O(h^{n+1})$, and hence the truncation error of the spatial operator $\mathcal{L}\mathbf{u} = \nabla \cdot (\mathbf{uu})$ should be $O(h^n)$. In Table 2, truncation errors of TLG discretization for $\mathcal{L}\mathbf{u}$ are listed with corresponding convergence rates; in both two and three dimensions, fourth-order and sixth-order convergence rates are clearly observed for TLG discretization with $n = 4$ and $n = 6$, respectively. This verifies the correctness and effectiveness of the TLG algorithm.

6.3 An elliptic equation with a cross-derivative term

In this test, our PLG-FD method is employed to numerically solve the elliptic equation

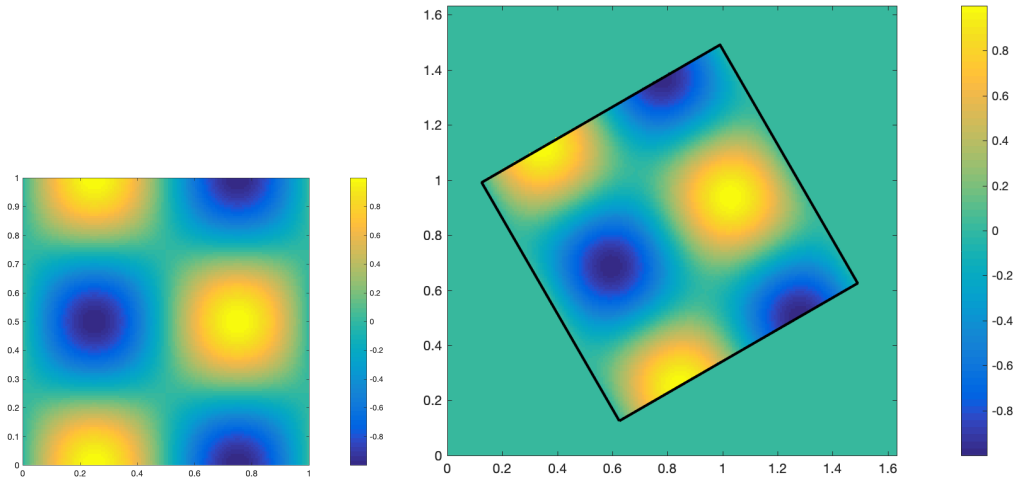
$$\begin{cases} a\dfrac{\partial^2 u}{\partial x^2} + b\dfrac{\partial^2 u}{\partial x \partial y} + c\dfrac{\partial^2 u}{\partial y^2} = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{87}$$

where Ω is a simply-connected domain and $u : \Omega \to \mathbb{R}$ is the unknown function. When b is not zero, the cross-derivative term could cause difficulty for FD methods based on one-dimensional FD formulas. In this test, the righthand side f and the boundary condition g are both derived from the exact solution

$$u(x, y) = \sin(2\pi x)\cos(2\pi y), \tag{88}$$

which is also used for calculating truncation and solution errors.

We design two test cases to examine how TLG discretization affects the accuracy of the numerical solutions. First, we solve (87) with $(a, b, c) = (1, 0, 2)$ on the unit square $[0, 1]^2$ and plot the solution in Figure 8 (a). The regular geometry and the absence of the cross-derivative term admit a discretization via one-dimensional FD formulas at all FD nodes, and hence TLG is not invoked and no TLG discretization is necessary. Second, we rotate the unit square by $\frac{\pi}{6}$ and embed it in a larger square domain, on which the TLG discretization is applied to irregular FD nodes for equation (87) with $(a, b, c) = (\frac{5}{4}, -\frac{\sqrt{3}}{2}, \frac{7}{4})$ and the resulting linear system is solved by a geometric multigrid method; the solution is plotted in Figure 8 (b). Since the exact solution is the same in both cases, the difference of the two computed solutions quantifies the influence of irregular geometry and cross-derivative terms.



(a) Solution on the unit square. (b) Solution on the unit square rotated by $\pi/6$.

Figure 8: Solutions of equation (87) by the PLG-FD method on different domains with the same grid size $h = \frac{1}{128}$. In subplot (a), the domain is the unit square where neither TLG nor TLG discretization is invoked; all discretizations are done via traditional one-dimensional FD formulas. In subplot (b), the domain is the rotated unit square where the TLG discretization is used for irregular FD nodes. See Table 3 for an error comparison.

Table 3: Truncation errors and solution errors of the PLG-FD method with $n = 4$ in solving the elliptic equation (87).

Truncation errors on the rotated square with $(a, b, c) = (\frac{5}{4}, -\frac{\sqrt{3}}{2}, \frac{7}{4})$							
	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$	rate	$h = \frac{1}{256}$	rate	$h = \frac{1}{512}$
L^∞	1.44e-01	2.58	2.42e-02	2.97	3.10e-03	3.36	3.01e-04
L^1	6.84e-04	4.02	4.22e-05	4.04	2.56e-06	4.02	1.57e-07
L^2	3.29e-03	3.60	2.71e-04	3.82	1.92e-05	3.85	1.33e-06
Solution errors on the rotated square with $(a, b, c) = (\frac{5}{4}, -\frac{\sqrt{3}}{2}, \frac{7}{4})$							
	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$	rate	$h = \frac{1}{256}$	rate	$h = \frac{1}{512}$
L^∞	2.02e-05	6.08	2.99e-07	4.22	1.61e-08	4.01	1.00e-09
L^1	1.28e-06	4.11	7.42e-08	4.02	4.58e-09	4.00	2.87e-10
L^2	1.75e-06	4.09	1.03e-07	4.01	6.38e-09	4.00	3.99e-10
Solution errors on the unit square $[0, 1]^2$ with $(a, b, c) = (1, 0, 2)$							
	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$	rate	$h = \frac{1}{256}$	rate	$h = \frac{1}{512}$
L^∞	1.24e-06	3.99	7.79e-08	4.00	4.88e-09	3.98	3.09e-10
L^1	3.30e-07	3.98	2.09e-08	3.99	1.31e-09	4.00	8.20e-11
L^2	4.81e-07	3.99	3.03e-08	3.99	1.90e-09	3.97	1.21e-10

As shown in Figure 8, the two solutions are qualitatively the same. In Table 3, we list truncation and solution errors of our fourth-order PLG-FD method in solving (87) on these two domains. For truncation errors, the convergence rates are above 3, 4, and 3.5 in the max-norm, the 1-norm, and the 2-norm, respectively. This implies that the elliptic operator is approximated to third-order accurate on the irregular FD nodes of codimension one and to fourth-order accurate on the regular FD nodes. In contrast, convergence rates of solution errors are very close to 4 in all norms, due to the elliptic nature of the PDE. The difference of solution errors on the two domains seems to suggest that TLG discretization, when dealing with complex geometry and cross-derivative terms, roughly triples the solution errors of traditional FD formulas.

6.4 Poisson's equation with mixed Dirichlet and Neumann conditions

In this subsection we demonstrate that our PLG-FD method can be much more accurate than a second-order embedded boundary (EB) method [14].

Consider a test [14, Problem 3] of Poisson's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \quad \text{in } \Omega, \tag{89}$$

where $\Omega = \Omega_1 \cap \Omega_2$, Ω_1 is the unit square centered at the origin and

$$\Omega_2 = \{(r, \theta) : r \geq 0.25 + 0.05 \cos 6\theta\};$$

a Dirichlet boundary condition is imposed on $\partial\Omega_1$ while a Neumann condition on $\partial\Omega_2$. Both f and the boundary conditions are derived from the exact solution

$$u(r, \theta) = r^4 \cos 3\theta, \tag{90}$$

where (r, θ) are polar coordinates satisfying $(x, y) = (r \cos\theta, r \sin\theta)$.

Although no cross-derivative term is present in the equation, TLG discretization is still necessary in order to fulfill the Neumann boundary condition near Ω_2; see step (c) of Definition 5.1.

As shown in Figure 9, solution errors of our PLG-FD method for $h = \frac{1}{160}$ are dominated by those near the curvilinear boundary. This is not unexpected because, while traditional FD discretizations on symmetric stencils have fortuitous cancellations of leading error terms, the TLG discretization near an irregular boundary does not. Therefore, truncation errors of the TLG discretization near the boundary are typically much higher than traditional FD discretization on regular FD nodes. Despite different types of discretization across the FD nodes, the solution errors appear smooth on the entire domain Ω.

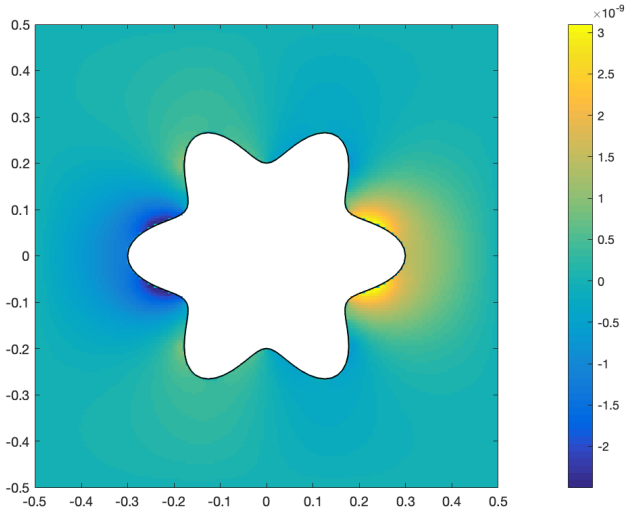


Figure 9: Solution errors of the fourth-order PLG-FD method in solving (89) with $h = \frac{1}{160}$. A Dirichlet condition and a Neumann condition are enforced on the square and the curvilinear boundary, respectively.

To compare the accuracy of our fourth-order method to the second-order EB method by Johansen and Colella [14], we show truncation errors and solution errors of these two methods in Table 4. The convergence rates of solution errors of our method are very close to four, meeting our expectations. Furthermore, our method is much more accurate than the EB method. In particular, the max-norm of solution errors of our method on the grid of $h = \frac{1}{80}$ is smaller by a factor of twenty than that of the EB method on the finest grid of $h = \frac{1}{320}$. Also, for $h = \frac{1}{320}$, the error norm of our method is much less than that of the EB method by a factor of 5350.

In Table 5, we report CPU time of the TLG discretization and that of all steps of our fourth-order PLG-FD method in solving (89). It is found that CPU time spent on TLG discretization is no more than 10% of that on the entire solution process; furthermore, this percentage decreases as the grid size is reduced. Invoked only at a set of codimension one, TLG discretization has a complexity of $O(\frac{1}{h^{\mathrm{D}-1}})$. In contrast, the complexity of multigrid solvers is $O(\frac{1}{h^{\mathrm{D}}})$. Hence the cost of TLG discretization is asymptotically negligible as far as CPU time of the entire solution process is concerned.

However, the discussion in the previous paragraph does not diminish the importance of test sets of type II: combined with Table 1, Table 5 also suggests that, on a fixed grid, CPU time of the entire PLG-FD method could have been dominated by that of TLG discretization if test sets of type I had been used.

Table 4: Errors of the fourth-order PLG-FD method and a second-order EB method [14] in solving Poisson's equation (89) with mixed Dirichlet and Neumann conditions.

Truncation errors of the EB method by Johansen and Colella [14]							
	$h = \frac{1}{40}$	rate	$h = \frac{1}{80}$	rate	$h = \frac{1}{160}$	rate	$h = \frac{1}{320}$
L^∞	1.66e-03	2.0	4.15e-04	2.0	1.04e-04	2.0	2.59e-05
Truncation errors of our fourth-order PLG-FD method							
	$h = \frac{1}{40}$	rate	$h = \frac{1}{80}$	rate	$h = \frac{1}{160}$	rate	$h = \frac{1}{320}$
L^∞	1.64e-03	4.57	6.91e-05	2.03	1.69e-05	2.82	2.39e-06
L^1	1.34e-05	4.27	6.94e-07	3.65	5.53e-08	4.07	3.30e-09
L^2	8.98e-05	4.51	3.95e-06	2.98	5.01e-07	3.56	4.24e-08
Solution errors of the EB method by Johansen and Colella [14]							
	$h = \frac{1}{40}$	rate	$h = \frac{1}{80}$	rate	$h = \frac{1}{160}$	rate	$h = \frac{1}{320}$
L^∞	4.78e-05	1.85	1.33e-05	1.98	3.37e-06	1.95	8.72e-07
Solution errors of our fourth-order PLG-FD method							
	$h = \frac{1}{40}$	rate	$h = \frac{1}{80}$	rate	$h = \frac{1}{160}$	rate	$h = \frac{1}{320}$
L^∞	4.37e-06	6.62	4.43e-08	4.00	2.76e-09	4.08	1.63e-10
L^1	4.16e-07	7.46	2.36e-09	2.94	3.07e-10	4.22	1.65e-11
L^2	7.59e-07	7.35	4.64e-09	3.26	4.84e-10	4.18	2.66e-11

Table 5: Timing results in solving Poisson's equation (89) using our fourth-order PLG-FD method on an Intel Xeon E5-2698 v3 at 2.30GHz. The middle two columns are CPU time in seconds.

h	TLG discretization	the entire solver	ratio
$\frac{1}{80}$	0.007	0.079	8.9%
$\frac{1}{160}$	0.013	0.277	4.7%
$\frac{1}{320}$	0.032	1.129	2.8%

Table 6: Errors of the fourth-order PLG-FD method in solving a three-dimensional Poisson's equation (91) with Dirichlet conditions.

Truncation errors							
	$h = \frac{1}{32}$	rate	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$	rate	$h = \frac{1}{256}$
L^∞	2.40e-01	2.50	4.24e-02	2.68	6.63e-03	3.01	8.23e-04
L^1	1.93e-03	3.85	1.34e-04	4.06	8.04e-06	4.01	4.98e-07
L^2	1.07e-02	3.18	1.18e-03	3.54	1.01e-04	3.48	9.06e-06
Solution errors							
	$h = \frac{1}{32}$	rate	$h = \frac{1}{64}$	rate	$h = \frac{1}{128}$		
L^∞	9.01e-05	4.09	5.30e-06	4.62	2.15e-07		
L^1	3.80e-06	4.40	1.80e-07	4.25	9.42e-09		
L^2	7.57e-06	4.54	3.26e-07	4.47	1.47e-08		

6.5 Poisson's equation in three dimensions

Consider a test of Poisson's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f \quad \text{in } \Omega', \tag{91}$$

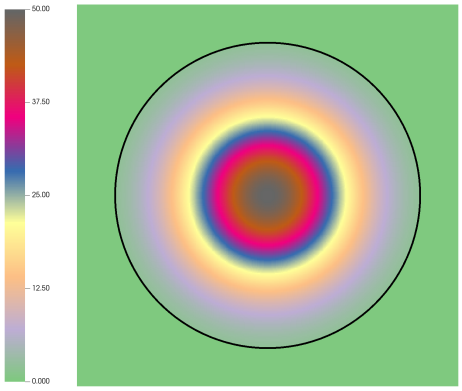
where $\Omega' = \Omega_3 \cap \Omega_4$, $\Omega_3 = [0,1]^3$ is the unit box and

$$\Omega_4 = \left\{ (x,y,z) \in \mathbb{R}^3 : \left(\frac{x-0.5}{0.25}\right)^2 + \left(\frac{y-0.5}{0.125}\right)^2 + \left(\frac{z-0.5}{0.25}\right)^2 \geq 1 \right\}. \tag{92}$$

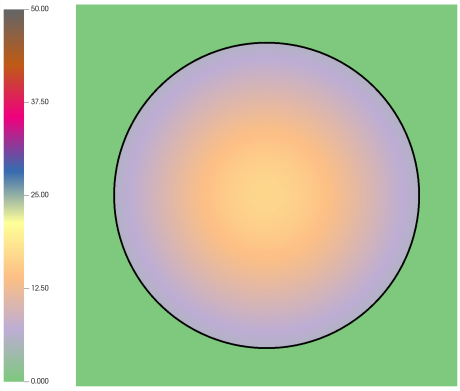
Dirichlet conditions are imposed on both $\partial\Omega_3$ and $\partial\Omega_4$. The source term f and the boundary conditions are derived from the exact solution

$$u(x,y,z) = \sin(2\pi x)\cos(2\pi y)\sin(2\pi z). \tag{93}$$

A variant of the three-dimensional TLG algorithm is used to discretize the Laplacian operator. Apart from the triangular lattice generated by the TLG algorithm, the FD nodes $\mathbf{i} \in \mathbb{Z}^D$ with $\|\mathbf{i} - \mathbf{q}\|_1 \leq 2$ where $\mathbf{q}$ is the starting node are also included in the discretization stencil. It is found that such stencils lead to substantially smaller solution error and fourth-order convergence; see Table 6.



(a) Solution on $T = 0.0$.



(b) Solution on $T = 1.0$.

Figure 10: Solutions of equation (94) by the PLG-FD method with the grid size $h = \frac{1}{160}$.

6.6 Solving heat equations with Robin boundary conditions

In this test, our PLG-FD method is employed to numerically solve the heat equation

$$\begin{cases} \dfrac{\partial \varphi}{\partial t} = \nu \Delta \varphi + f \quad \text{in } \Omega, \\ \nu \dfrac{\partial \varphi}{\partial \mathbf{n}} + a\varphi = g \quad \text{on } \partial\Omega, \end{cases} \tag{94}$$

where Ω is a simply-connected domain, φ is the unknown function, f, g, a are given functions, ν is the diffusion coefficient, and $\mathbf{n}$ is the outward unit normal of $\partial\Omega$.

We consider the diffusion of a point source within a disk of radius R. The exact solution is

$$\varphi(x, y, t) = \frac{10}{4\nu(t + \frac{1}{2})} e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{4\nu(t+\frac{1}{2})}}, \tag{95}$$

where (x_0, y_0) is the center of the disk. We set $a = 1$ and use the method of manufactured solutions to determine the value of $g(x, y, t)$. The radius and center of the disk we use are $R = 0.8$ and $(x_0, y_0) = (1, 1)$. The diffusion coefficient is 0.1. We use the classic fourth-order Runge–Kutta scheme and the fourth-order PLG-FD discretization. The time step is taken as $\Delta t = \frac{h^2}{6.25\nu}$. Figure 10 shows the initial and final numerical solutions with the grid size $h = \frac{1}{160}$. The errors and convergence rates at the final time $T = 1.0$ are listed in Table 7. The convergence rates of solution errors of our method exceed four in the L^1, L^2, and L^∞ norms.

Table 7: Errors of the fourth-order PLG-FD method in solving a heat equation (94) with Robin boundary conditions.

	$h=\frac{1}{20}$	rate	$h=\frac{1}{40}$	rate	$h=\frac{1}{80}$	rate	$h=\frac{1}{160}$
L^∞	5.36e-04	4.48	2.40e-05	4.12	1.38e-06	4.09	8.13e-08
L^1	3.35e-04	4.48	1.50e-05	4.08	8.90e-07	4.11	5.20e-08
L^2	2.55e-04	4.50	1.13e-05	4.09	6.65e-07	4.11	3.86e-08

6.7 Solving advection-diffusion equations with Neumann boundary conditions

In this subsection, we consider the advection-diffusion equation

$$\frac{\partial \varphi}{\partial t} + \mathbf{u} \cdot \nabla \varphi = \nu \Delta \varphi \quad \text{in } \Omega, \tag{96}$$

where φ is the unknown function, $\mathbf{u}(x, y, t)$ is the specified velocity field, and ν is the diffusion coefficient. We apply Neumann boundary conditions of the form

$$\frac{\partial \varphi}{\partial \mathbf{n}} = 0. \tag{97}$$

We consider the diffusion and solid body rotation inside an annulus composed of two concentric circles. The radius of the inner circle of the annulus is $R_1 = 0.25$. The outer radius is $R_2 = 0.8$. The center of each circle is at $(1.0, 1.0)$. The diffusion coefficient is $\nu = 0.01$. The initial profile of our solution is a function of θ in the form of

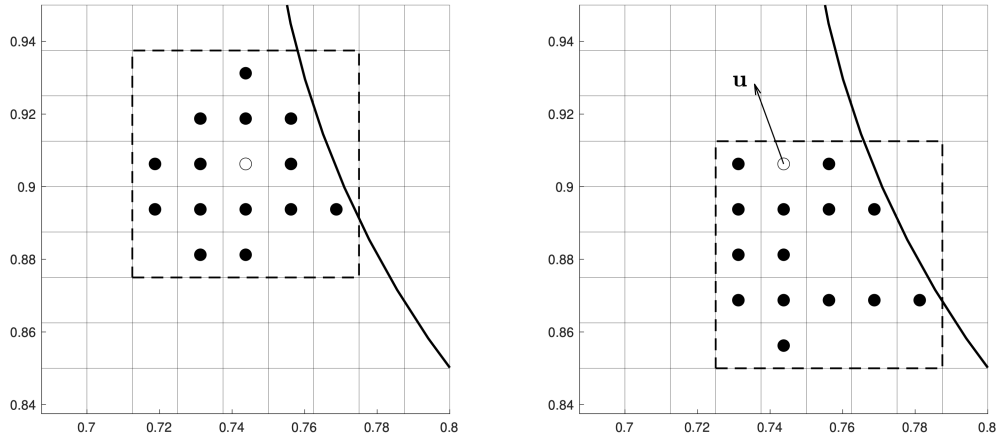
$$\varphi(\theta) = 0.5 \left(\operatorname{erf}\left(\frac{\pi/6 - \theta}{\sqrt{4\nu}} \right) + \operatorname{erf}\left(\frac{\pi/6 + \theta}{\sqrt{4\nu}} \right) \right), \tag{98}$$

where $\theta(x, y)$ is the angle between the vector $(x - 1, y - 1)$ and the x-axis. The velocities for solid body rotation can be determined from the stream function

$$\psi(x, y) = \begin{cases} 0, & \text{if } r \in [0, R_1) \cup (R_2, \infty), \\ 0.2\pi(R_2^2 - r^2), & \text{if } r \in [R_1, R_2], \end{cases} \tag{99}$$

where $r = \sqrt{(x-1)^2 + (y-1)^2}$ is the distance from the point (x, y) to the center of the circle.

We use the classic fourth-order Runge–Kutta scheme and the fourth-order PLG-FD discretization. We use different strategies to discretize the Laplacian and the convection operator. For the Laplacian term, we use a compact scheme, whereas for the convection term, we use an upwind scheme. See Figure 11 for an illustration. The time step is taken as $\Delta t = \frac{h^2}{3.9\nu}$. Figure 12 shows the results of solid body rotation coupled with diffusion in the



(a) Discretizing $\Delta\varphi$.

(b) Discretizing $\mathbf{u} \cdot \nabla\varphi$.

Figure 11: Examples of discretizing operators using different strategies in Section 5.1. In all cases, the starting point $\mathbf{q}$ is represented by a hollow dot near the boundary of the inner circle, and all the solid and hollow dots form a poised lattice.

Table 8: Errors of the fourth-order PLG-FD method in solving an advection-diffusion equation (96) with Neumann boundary conditions.

	$h = \frac{1}{80} - \frac{1}{160}$	rate	$h = \frac{1}{160} - \frac{1}{320}$	rate	$h = \frac{1}{320} - \frac{1}{640}$
L^∞	5.91e-05	3.45	5.43e-06	3.62	4.40e-07
L^1	1.54e-06	3.79	1.12e-07	3.69	8.67e-09
L^2	3.23e-06	3.73	2.44e-07	3.60	2.01e-08

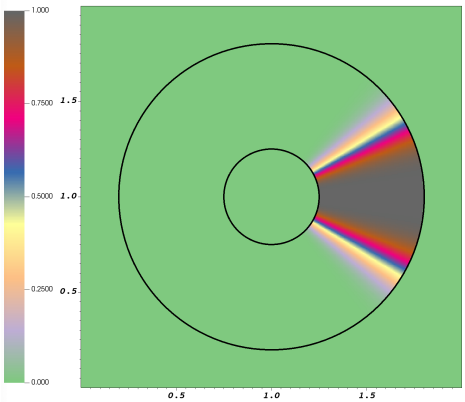
annulus at various times over one full rotation. The errors and convergence rates at time $T = 1.0$ are listed in Table 8. The convergence rates of solution errors of our method are close to four, meeting our expectations.

7 Conclusion

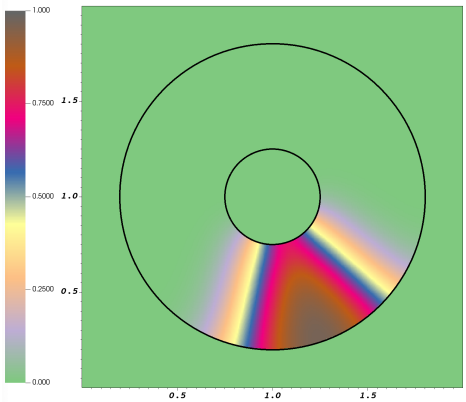
For a positive integer n and a given set K of isolated points in $\mathbb{Z}^{\mathsf{D}}$, we propose a generic TLG algorithm that outputs a subset $\mathcal{T} \subset K$ such that multivariate polynomial interpolation on $\mathcal{T}$ is unisolvent in Π_n^{D}.

Based on the TLG discretization, we further develop a fourth-order PLG-FD method for numerically solving PDEs on irregular domains with Cartesian grids. Our new method has been shown to be simple, efficient, and more accurate than a second-order EB method.

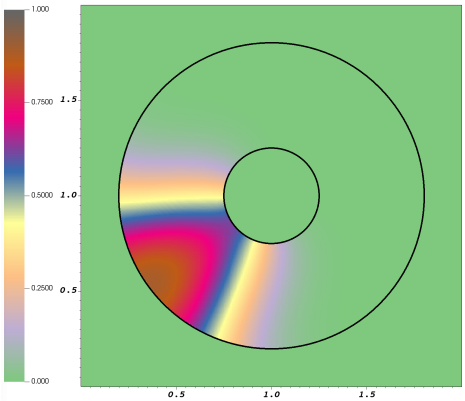
Several research prospects follow. We are currently working on augmenting the PLG-FD method to higher convergence rates and to time-dependent



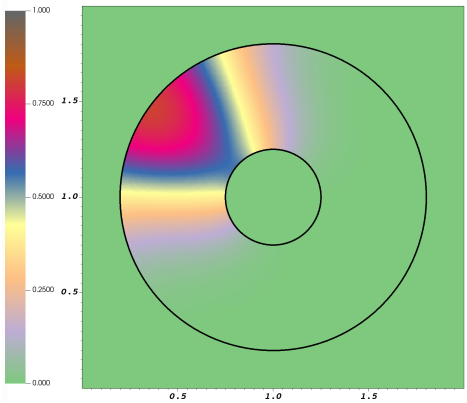
(a) Solution on $T = 0.0$.



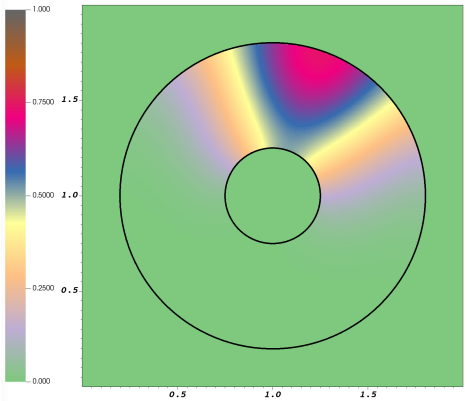
(b) Solution on $T = 1.0$.



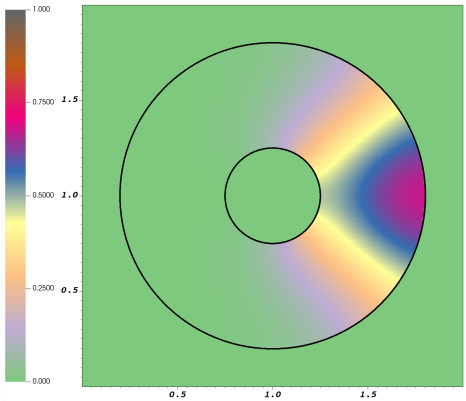
(c) Solution on $T = 2.0$.



(d) Solution on $T = 3.0$.



(e) Solution on $T = 4.0$.



(f) Solution on $T = 5.0$.

Figure 12: Solutions of equation (96) by the PLG-FD method with the grid size $h = \frac{1}{160}$.

problems such as the Navier-Stokes equations on irregular domains. We also plan to utilize the TLG discretization in our finite volume methods [38] for simulating incompressible fluids with moving boundaries.

References

[1] J. M. Carnicer, M. Gasca, and T. Sauer. Interpolation lattices in several variables. *Numer. Math.*, 102:559–581, 2006.

[2] J. M. Carnicer, M. Gasca, and T. Sauer. Aitken-Neville sets, principal lattices and divided differences. *J. Approx. Theory*, 156(2):154–172, 2009.

[3] J. M. Carnicer and C. Godes. Geometric characterization and generalized principal lattices. *J. Approx. Theory*, 143:2–14, 2006.

[4] K. C. Chung and T. H. Yao. On lattices admitting unique Lagrange interpolation. *SIAM J. Numer. Anal.*, 14(4):735–743, 1977.

[5] C. de Boor. Multivariate polynomial interpolation: Aitken-Neville sets and generalized principal lattices. *J. Approx. Theory*, 161:411–420, 2009.

[6] D. Devendran, D. T. Graves, H. Johansen, and T. Ligocki. A fourth-order Cartesian grid embedded boundary method for Poisson's equation. *Commun. Appl. Math. Comput. Sci.*, 12:51–79, 2017.

[7] N. Dyn and M. S. Floater. Multivariate polynomial interpolation on lower sets. *J. Approx. Theory*, 177:34–42, 2014.

[8] M. Errachid, A. Essanhaji, and A. Messaoudi. RMVPIA: a new algorithm for computing the Lagrange multivariate polynomial interpolation. *J. Sci. Comput.*, 84:1507–1534, 2020.

[9] M. Gasca and T. Sauer. Polynomial interpolation in several variables. *Adv. Comput. Math.*, 12:377–410, 2000.

[10] W. Gautschi. *Numerical Analysis*. Birkhauser, second edition, 2012. ISBN: 978-0-8176-8258-3.

[11] D. M. Ingram, D. M. Causon, and C. G. Mingham. Developments in Cartesian cut cell methods. *Mathematics and Computers in Simulation*, 61:561–572, 2003.

[12] K. Ito, Z. Li, and Y. Kyei. Higher-order, Cartesian grid based finite difference schemes for elliptic equations on irregular domains. *SIAM J Sci. Comput.*, 27(1):346–367, 2005.

[13] G. Jaklic, J. Kozak, M. Krajnc, V. Vitrih, and E. Zagar. Lattices on simplicial partitions. *J. Comput. Appl. Math.*, 233:1704–1715, 2010.

[14] H. Johansen and P. Colella. A cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 147(1):60–85, 1998.

[15] S. L. Lee and G. M. Phillips. Construction of lattices for Lagrange interpolation in projective space. *Constr. Approx.*, 7:283–297, 1991.

[16] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations.* SIAM, Philadelphia PA, 2007.

[17] R. J. Leveque and Z. Li. Immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM J. on Numer. Anal.*, 31(4):1019–1044, 1994.

[18] Z. Li and K. Ito. *The Immersed Interface Method: Numerical Solutions of PDEs Involving Interfaces and Irregular Domains.* SIAM, 2006.

[19] Z. Li, Z. Qiao, and T. Tang. *Numerical Solution of Differential Equations: Introduction to Finite Difference and Finite Element Methods.* Cambridge University Press, Cambridge, United Kingdom, 2018.

[20] T. G. Liu, B. C. Khoo, and K. S. Yeo. Ghost fluid method for strong shock impacting on material interface. *J. Comput. Phys.*, 190(2):651–681, 2003.

[21] A. Mayo. The fast solution of Poisson's and the biharmonic equations on irregular regions. *SIAM J. Numer. Anal.*, 21:285–299, 1984.

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

[23] R. D. Neidinger. Multivariate polynomial interpolation in Newton forms. *SIAM Review*, 61(2):361–381, 2019.

[24] N. Overton-Katz, X. Gao, S. Guzik, O. Antepara, D. T. Graves, and H. Johansen. A fourth-order embedded boundary finite volume method for the unsteady Stokes equations with complex geometries. *SIAM J. Sci. Comput.*, 45(5):A2409–A2430, 2023.

[25] C. S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[26] G. M. Phillips. *Interpolation and Approximation by Polynomials.* Springer, 2003.

[27] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2020. ISBN: 978-0134610993.

[28] T. Sauer. Lagrange interpolation on subgrids of tensor product grids. *Math. Comput.*, 73(245):181–190, 2003.

[29] T. Sauer and Y. Xu. On multivariate Lagrange interpolation. *Math. Comput.*, 64(211):1147–1170, 1995.

[30] T. Sauer and Y. Xu. The Aitken-Neville scheme in several variables. In *Approximation Theory X: Abstract and Classical Analysis*, pages 353–366, Nashville, 2002. Vanderbilt University Press.

[31] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Wadsworth & Brooks, Belmont CA, 1989.

[32] L. N. Trefethen. *Approximation Theory And Approximation Practice*. SIAM, 2017. ISBN: 978-9386235442.

[33] Y.-H. Tseng and J. H. Ferziger. A ghost-cell immersed boundary method for flow in complex geometry. *J. Comput. Phys.*, 192:593–623, 2003.

[34] P. G. Tucker and Z. Pan. A Cartesian cut cell method for incompressible viscous flow. *Applied Mathematical Modelling*, 24(8-9):591–606, 2000.

[35] H. Werner. Remarks on Newton type multivariate interpolation for subsets of grids. *Comput.*, pages 181–191, 1980.

[36] Y. Xie and W. Ying. A fourth-order kernel-free boundary integral method for implicitly defined surfaces in three space dimensions. *J. Comput. Phys.*, 415:109526, 2020.

[37] L. Xu and T. G. Liu. Ghost-fluid-based sharp interface methods for multi-material dynamics: a review. *Commun. Comput. Phys.*, 34(3):563–612, 2023.

[38] Q. Zhang. GePUP: Generic projection and unconstrained PPE for fourth-order solutions of the incompressible Navier-Stokes equations with no-slip boundary conditions. *J. Sci. Comput.*, 67:1134–1180, 2016.

[39] Q. Zhang and Z. Li. Boolean algebra of two-dimensional continua with arbitrarily complex topology. *Math. Comput.*, 89:2333–2364, 2020.

[40] Q. Zhang and P. L.-F. Liu. Handling solid-fluid interfaces for viscous flows: explicit jump approximation vs. ghost cell approaches. *J. Comput. Phys.*, 229:4225–46, 2010.

[41] Y. C. Zhou, S. Zhao, M. Feig, and G. W. Wei. High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources. *J. Comput. Phys.*, 213:1–30, 2006.