Efficient discretization of the Laplacian: application to moving boundary problems

Sebastian Josue Castillo
Advisor: Dr. Ferenc Izsák

Eötvös Loránd University, Budapest

December 2024

Outline

Problem statement

Let $\Omega \subset \mathbb{R}^2$. We consider

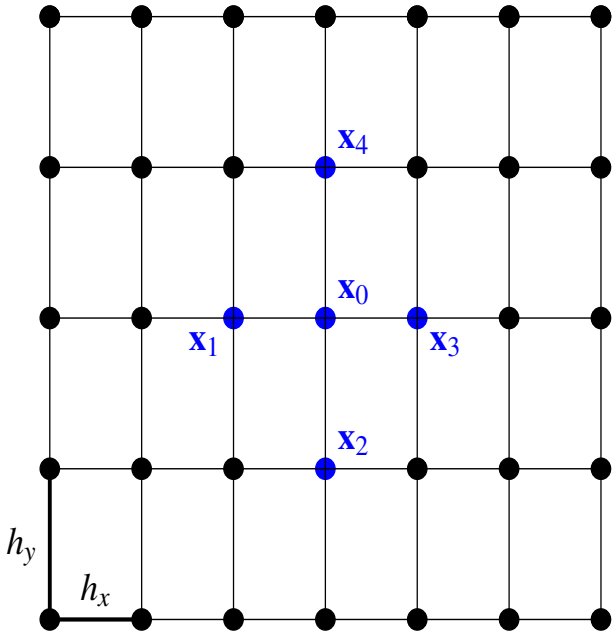
$$\begin{cases} \Delta u(\mathbf{x}) = f(\mathbf{x}) & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = g(\mathbf{x}) & \mathbf{x} \in \partial\Omega, \end{cases} \tag{1.1}$$

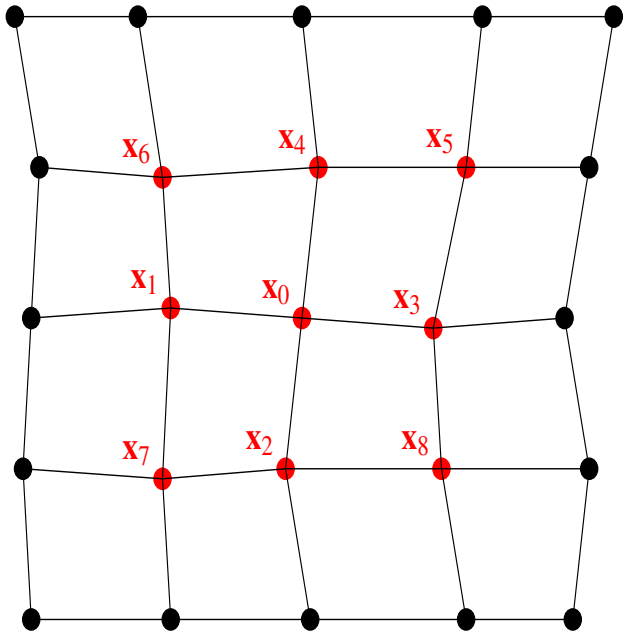
where $f \in \mathrm{L}^2(\Omega)$ and $g \in C(\partial\Omega)$ are given.

Goal

To approximate the Laplacian of a function $u \in C^2(\Omega)$ in $\mathbf{x}_0 \in \Omega$ based on the corresponding function values in its neighboring grid points:

$$\Delta u(\mathbf{x}_0) \approx \sum_{j=0}^{k} a_j u(\mathbf{x}_j). \tag{1.2}$$





- Approximation with the left geometry:

$$\Delta u(\mathbf{x}_0) \approx -(\frac{2}{h_x^2}+\frac{2}{h_y^2})u(\mathbf{x}_0)+\frac{1}{h_x^2}u(\mathbf{x}_1)+\frac{1}{h_x^2}u(\mathbf{x}_3)+\frac{1}{h_y^2}u(\mathbf{x}_2)+\frac{1}{h_y^2}u(\mathbf{x}_4) \tag{1.3}$$

  with an error $\mathcal{O}(h_x^2) + \mathcal{O}(h_y^2)$.
- For a general geometry, this accuracy is lost.

Objective

To find the coefficients in (1.3) for a general mesh using an **optimization procedure**.

Pointwise optimization

9 grid point approximation

$$\Delta u(\mathbf{x}_0) \approx \sum_{j=0}^{9} a_j u(\mathbf{x}_j). \tag{2.4}$$

1. **Uniform mesh with grid size h:** to obtain an approximation order $N-1$, the equality

$$\Delta p(\mathbf{x}_0) = a_0 p(\mathbf{x}_0) + a_1 p(\mathbf{x}_1) + \cdots + a_8 p(\mathbf{x}_8). \tag{2.5}$$

   should hold for all polynomials p up to order N (coefficients are scaled as h^{-2}).

2. **General mesh geometries:** The approximation order is decreased and (2.5) is not valid anymore for all polynomials up to order N.

What polynomials should we consider in

$$\Delta p(\mathbf{x}_0) = a_0 p(\mathbf{x}_0) + a_1 p(\mathbf{x}_1) + \cdots + a_8 p(\mathbf{x}_8).$$

- Constant polynomials should satisfy (2.4) $\Longrightarrow$ Constraint:

$$a_0 = -(a_1 + \cdots + a_8).$$

- Polynomials that span the space of first and second-order polynomials:

$$\begin{aligned} &p_1(x,y) = 100x,\ p_2(x,y) = 100y,\ p_3(x,y) = 10xy, \\ &p_4(x,y) = 10x^2,\ p_5(x,y) = 10(x^2 - y^2), \end{aligned} \tag{2.6}$$

- Weights are important for accuracy.

- Additionally, we take harmonic polynomials up to order five:

$$p_6(x,y) = x^3 - 3x^2y, \quad p_7(x,y) = y^3 - 3xy^2,$$
$$p_8(x,y) = x^3y - xy^3, \quad p_9(x,y) = x^4 + 6x^2y^2 - y^4,$$
$$p_{10}(x,y) = 5x^4y - 10x^2y^3 + y^5, \quad p_{11}(x,y) = x^5 - 10x^3y^2 + 5xy^4,$$

Approximation of $\Delta p_j(\mathbf{x}_0)$:

$$a_1 p_j(\mathbf{x}_1) + a_2 p_j(\mathbf{x}_2) + \cdots + a_8 p_j(\mathbf{x}_8) \approx \Delta p_j(\mathbf{x}_0), \quad j = 1, 2, \ldots, 11 \tag{2.7}$$

for the given geometrical setup.

Optimization procedure

- Find the vector of coefficients $\mathbf{a} = [a_1, a_2, \ldots, a_8]^T$ in (2.7) such that $P \cdot \mathbf{a} \approx \Delta\mathbf{p}$, where

$\rightarrow$ $P \in \mathbf{R}^{11\times 8}$ with $P_{jk} = p_j(\mathbf{x}_k)$;

$\rightarrow$ $\Delta\mathbf{p} = [0, 0, 0, 20, 0, 0, 0, 0, 0, 0, 0]^T \in \mathbf{R}^{11}$, so that $\Delta\mathbf{p}_j = p_j(\mathbf{x}_0)$.

The optimization task is solved in the least-square sense for each point $\mathbf{x}_0$ using the `np.linalg.lstsq` subroutine in Python.

Vectorized procedure

1. We create a matrix, where each row contains the geometry around a fixed grid point.

2. The optimization procedure is applied (independently) to the rows of this matrix using the `np.apply_along_axis` subroutine in Python.

Model problem

Test problem:

$$\begin{cases} \Delta u(x,y) = -16x\cos 4y & (x,y) \in \Omega \\ u(x,y) = u_g(x,y) & (x,y) \in \partial\Omega, \end{cases} \tag{3.8}$$

with the computational domain

$$\Omega = \{(x,y) \in \mathbb{R}^2 : 0 < x < \frac{\pi}{2}, \quad 0 < y < 1 + \frac{\cos x}{4}\}$$

such that

- Left and right side: $u_g(0,y) = y$ and $u_g(\frac{\pi}{2},y) = y + \frac{\pi}{2} \cdot \cos 4y$.
- Bottom side: $u_g(x,0) = x$.
- Top side: $u(x, 1 + \frac{\cos x}{4}) = 1 + \frac{\cos x}{4} + x \cdot \cos 4(1 + \frac{\cos x}{4})$.

Analytic solution is $u(x,y) = y + x \cdot \cos 4y$.

Spatial discretization: grid space is refined in the vicinity of the top boundary.

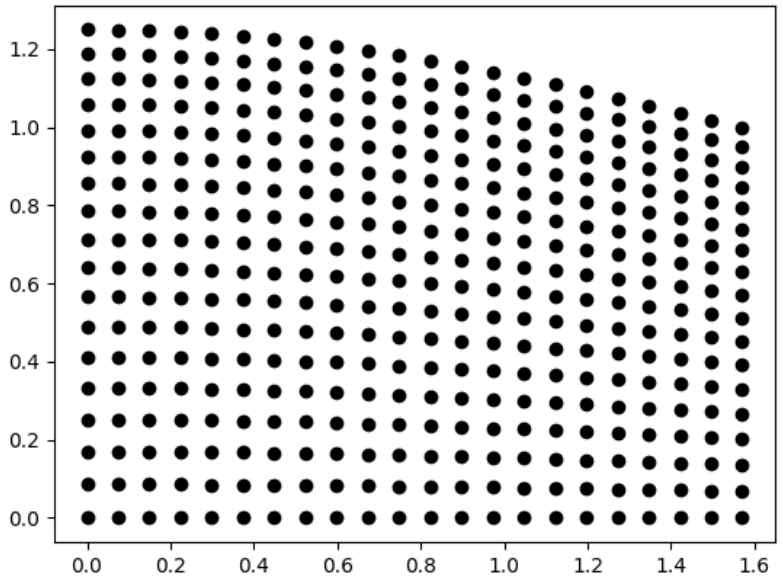


Figure: Number of uniformly distributed grid points: 20 in the horizontal direction and 16 in the vertical direction.

Table: Simulation time (ms) and discrete L_2-norm error for different discretization parameters in the model problem

N_x	10	15	20	25	30	35	40	45	50	55
N_y	8	12	16	20	24	28	32	36	40	44
time (ms)	22	20	27	41	55	73	95	119	146	174
error $\cdot 10^3$	10.9	5.2	3.1	2.1	1.6	1.2	1	0.87	0.77	0.7

Stefan problem

- Stefan problems can be used to describe the interaction of different phases of materials.
- We consider $u : \Omega \to \mathbb{R}$ unknown function corresponding to the temperature in a melting system (ice and water).

At each time $t \in [0, T]$, the domain Ω is separated into the two disjoint ones

$$\underbrace{\Omega_{0,t} = \{\mathbf{x} \in \Omega : u(t, \mathbf{x}) < 0\}}_{ice}; \text{ and } \underbrace{\Omega_{1,t} = \{\mathbf{x} \in \Omega : u(t, \mathbf{x}) > 0\}}_{water},$$

so that $\overline{\Omega} = \overline{\Omega_{0,t}} \cup \overline{\Omega_{1,t}}$.

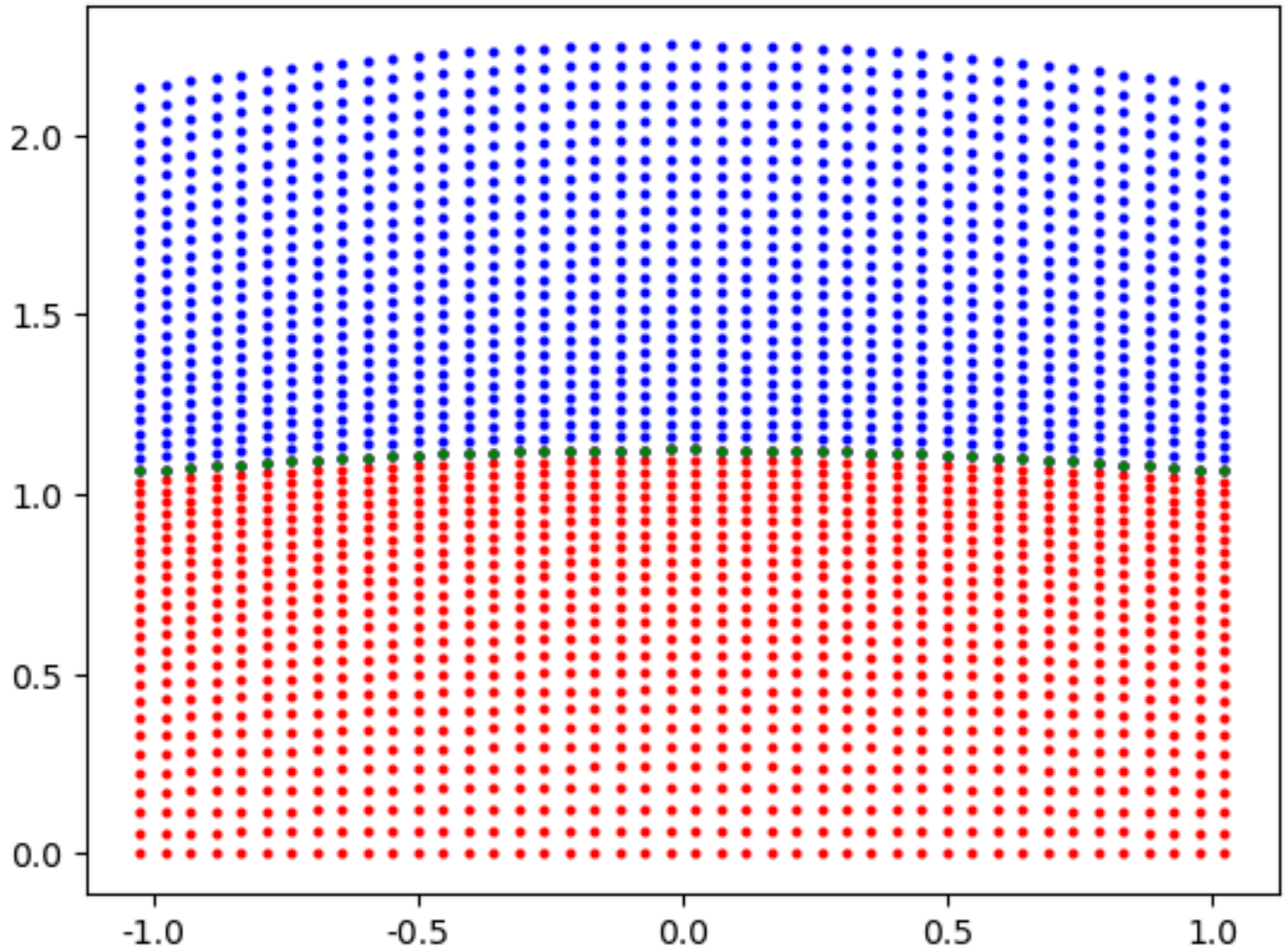


Figure: Initial grid.

- Common boundary:

$$\Gamma_t = \{\mathbf{x} = (x(t), y(t)) \in \overline{\Omega} : u(t, \mathbf{x}) = 0\}.$$

- Notation:

$$u_0 = u|_{[0,T]\times\Omega_{0,t}} \text{ and } u_1 = u|_{[0,T]\times\Omega_{1,t}}.$$

Governing equations

$$\begin{cases} \partial_t u_0(t, \mathbf{x}) = D_0 \cdot \Delta u_0(t, \mathbf{x}), & \mathbf{x} \in \Omega_{0,t} \\ \partial_t u_1(t, \mathbf{x}) = D_1 \cdot \Delta u_1(t, \mathbf{x}), & \mathbf{x} \in \Omega_{1,t}. \end{cases} \tag{3.9}$$

Evolution of the common boundary

$$\partial_t \Gamma_t(\mathbf{x}) = -L_H \cdot [[\partial_\nu u(t, x(t), y(t))]], \tag{3.10}$$

- Γ_t level set parametrized with t.
- L_H latent heat of solidification.
- Jump operator $[[\cdot]]$, defined on the common boundary Γ_t with

$$[[\partial_\nu u(t, \mathbf{x})]] = D_0 \cdot \nu_0(\mathbf{x}) \cdot \nabla u_0(t, \mathbf{x}) + D_1 \cdot \nu_1(\mathbf{x}) \cdot \nabla u_1(t, \mathbf{x}), \tag{3.11}$$

- ν_0, ν_1 corresponding (opposite) outward normals.

Initial and boundary conditions

- Initial condition:

$$u(0,x,y) = 1 + \frac{\cos x}{4} - y.$$

- Top boundary condition:

$$u(t,x,2+\frac{\cos x}{2}) = -1 + (\frac{t}{4} - 1)\frac{\cos x}{4}, \quad x \in [-1,1]. \qquad (3.12)$$

- Bottom boundary condition:

$$u(t,x,0) = 1 + (t+1)\cdot\frac{\cos x}{4}, \quad x \in [-1,1]. \qquad (3.13)$$

- Right and left boundary conditions: homogeneous Neumann-type.

Construction of the grid

We apply a structured quadrilateral grid.

- Mesh only moving vertically $\Longrightarrow$ fix the x coordinates of the grid points $x_1, x_2, \ldots, x_{Nx}$. Horizontal spacing $hx = 2/(Nx+1)$.
- Neumann boundary conditions on the left and right are dealt with ghost grid points.
- We fix the number of grid points in the vertical direction: $Ny/2$ grid points both in Ω_0 and Ω_1.

In the vicinity of the interface Γ_t, we apply a finer grid $\Longrightarrow$ more accurate approximation of the evolution of Γ_t.

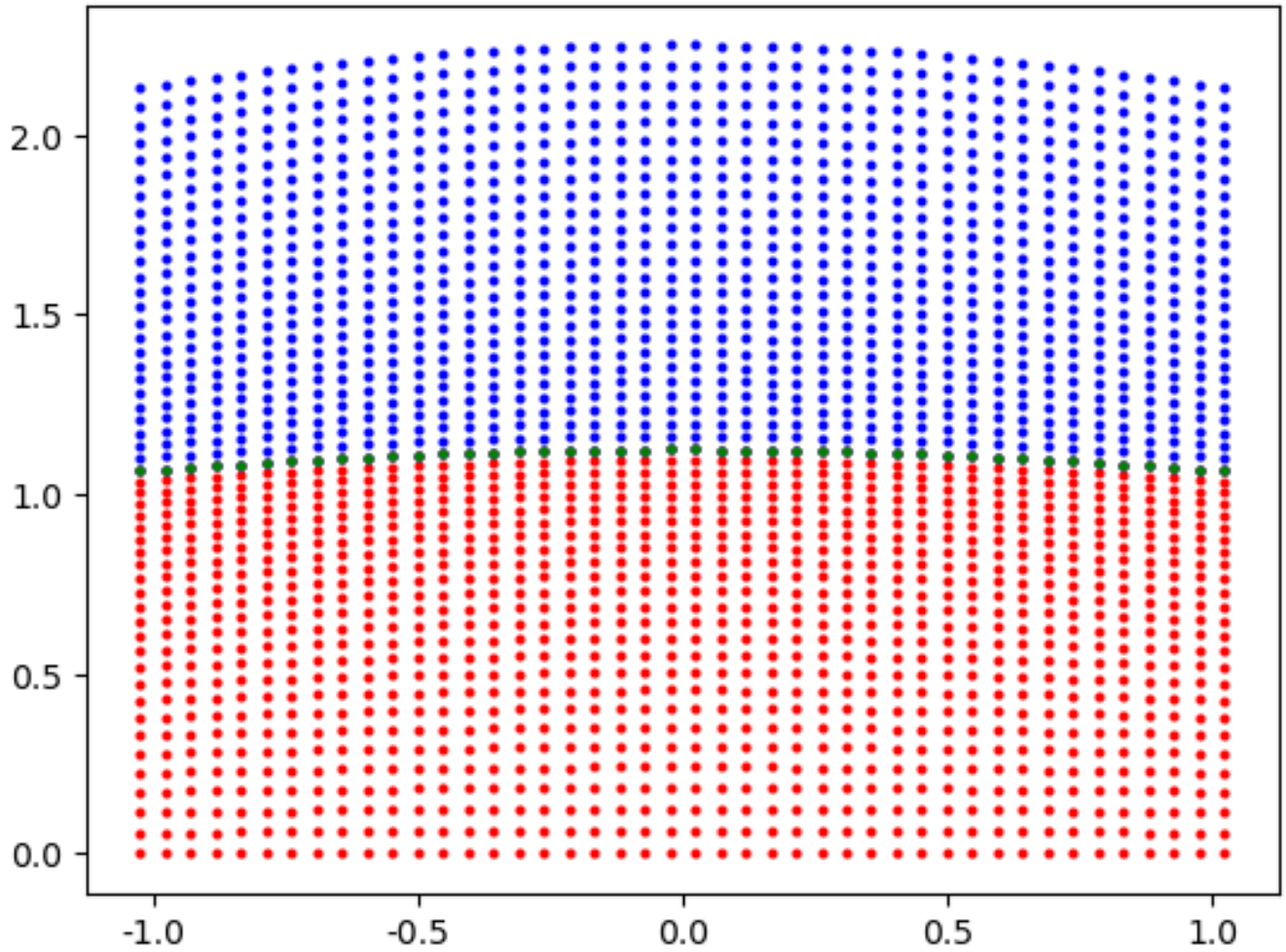


Figure: Initial grid.

Sketch of algorithm

(i) Diffusion problem on top domain Ω_0 and bottom domain Ω_1 with some initial data.

(a) We construct the discretization of the Laplacian using the pointwise optimization algorithm.

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \delta \cdot (\Delta_h \mathbf{u}^{n+1} + \mathbf{f}^{n+1}), \tag{3.14}$$

(b) We incorporate the Dirichlet- and Neumann-type boundary conditions:

$$(I - \delta \cdot (\tilde{\Delta}_h))\mathbf{u}^{n+1} = \mathbf{u}^n + \delta \cdot \tilde{\mathbf{f}}^{n+1}, \tag{3.15}$$

(c) We perform a time step by solving (3.15).

(ii) New Neumann boundary data on the moving surface.

(a) We move the surface according to equation (3.10).

$\rightarrow$ At each point $(x(t), y(t)) \in \Gamma_t$ we perform

$$\begin{aligned}(x((n+1)\delta), y((n+1)\delta)) &\approx (x(n\delta), y(n\delta)) + \delta \cdot \partial_t \Gamma_{n\delta}(x(n\delta), y(n\delta)) \\ &= (x(n\delta), y(n\delta)) + \delta \cdot [[\partial_{\boldsymbol{\nu}} u(x(n\delta), y(n\delta))]],\end{aligned} \tag{3.16}$$

We move segments of the common boundary by computing

$$\mathbf{d}_i = \delta \cdot (\boldsymbol{\nu}_i \cdot (\nabla \mathbf{u}_0^n - \nabla \mathbf{u}_1^n))\boldsymbol{\nu}_i$$

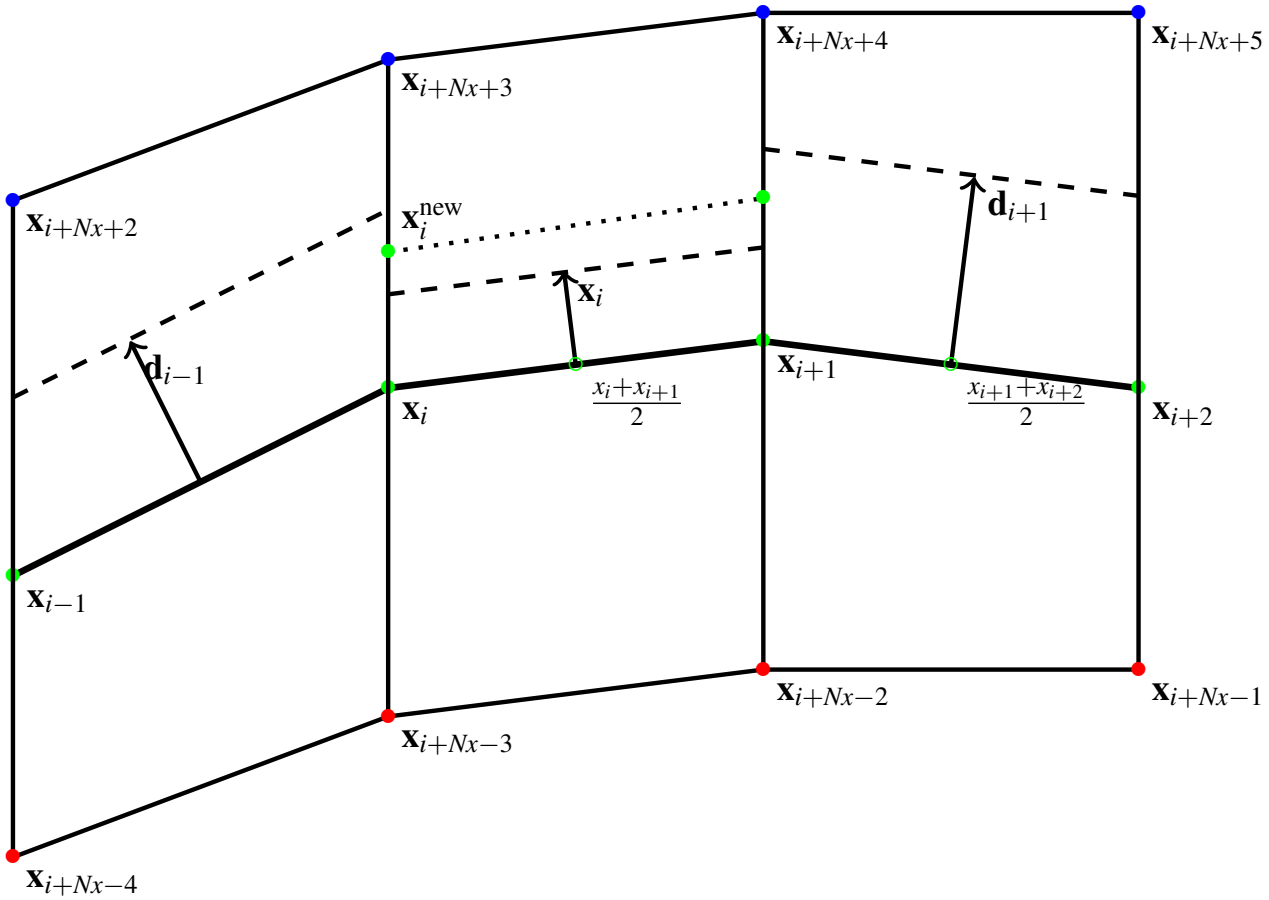


Figure: Motion of the segments $\mathbf{x}_{i-1}\mathbf{x}_i$, $\mathbf{x}_i\mathbf{x}_{i+1}$, and $\mathbf{x}_{i+1}\mathbf{x}_{i+2}$ of Γ^n in step (ii) (a). d_i denotes the shift vectors obtained from the jump term.

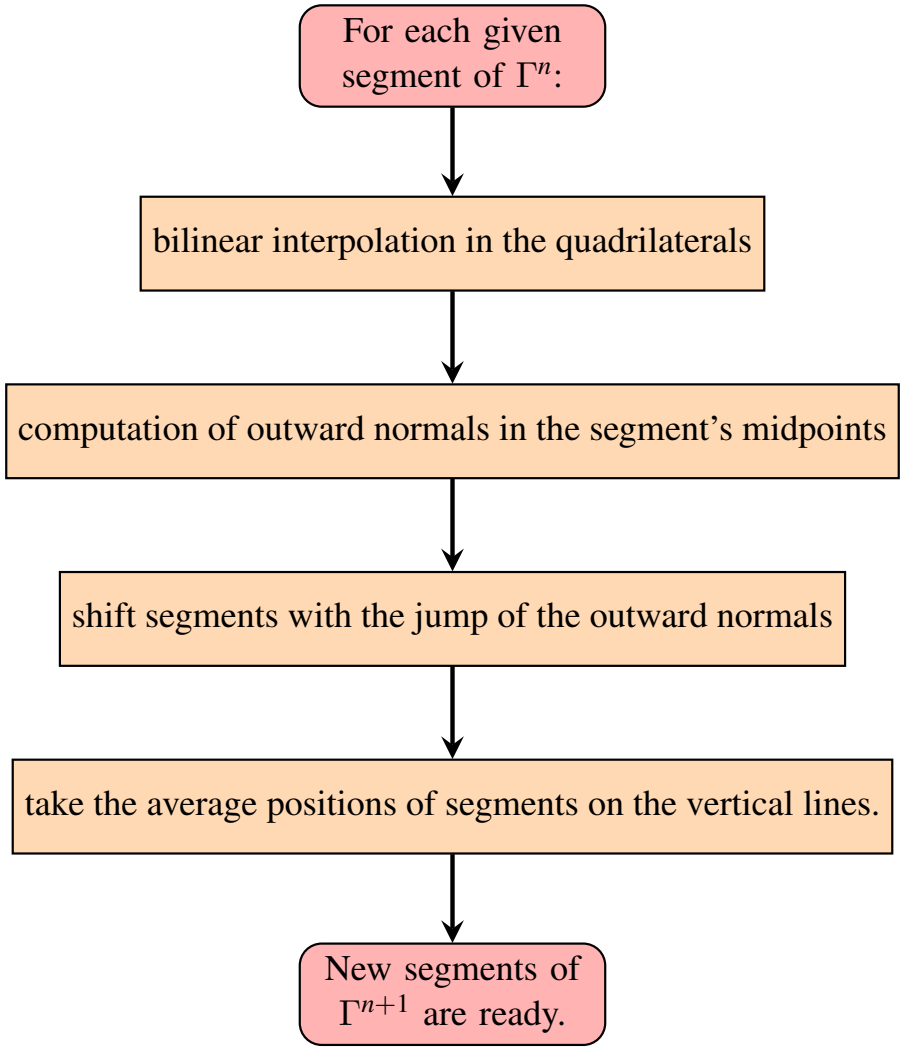


Figure: The steps of shifting the interface Γ^n to obtain the new one Γ^{n+1}.

(b) We interpolate the solutions on the two new domains.

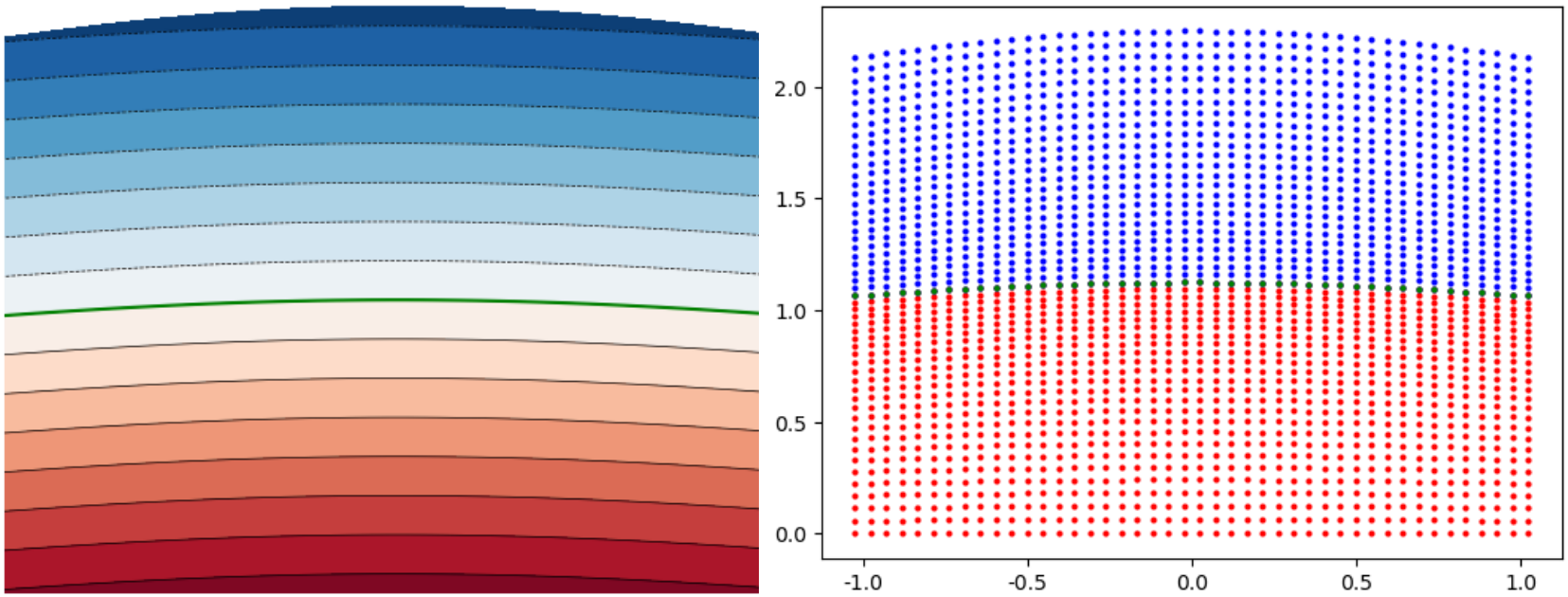


Figure: Initial condition (**left**, with a heat map) and the corresponding initial grid (**right**) for the simulation of the problem in (3.9)–(3.10). The common interface and the corresponding grid points are colored green.

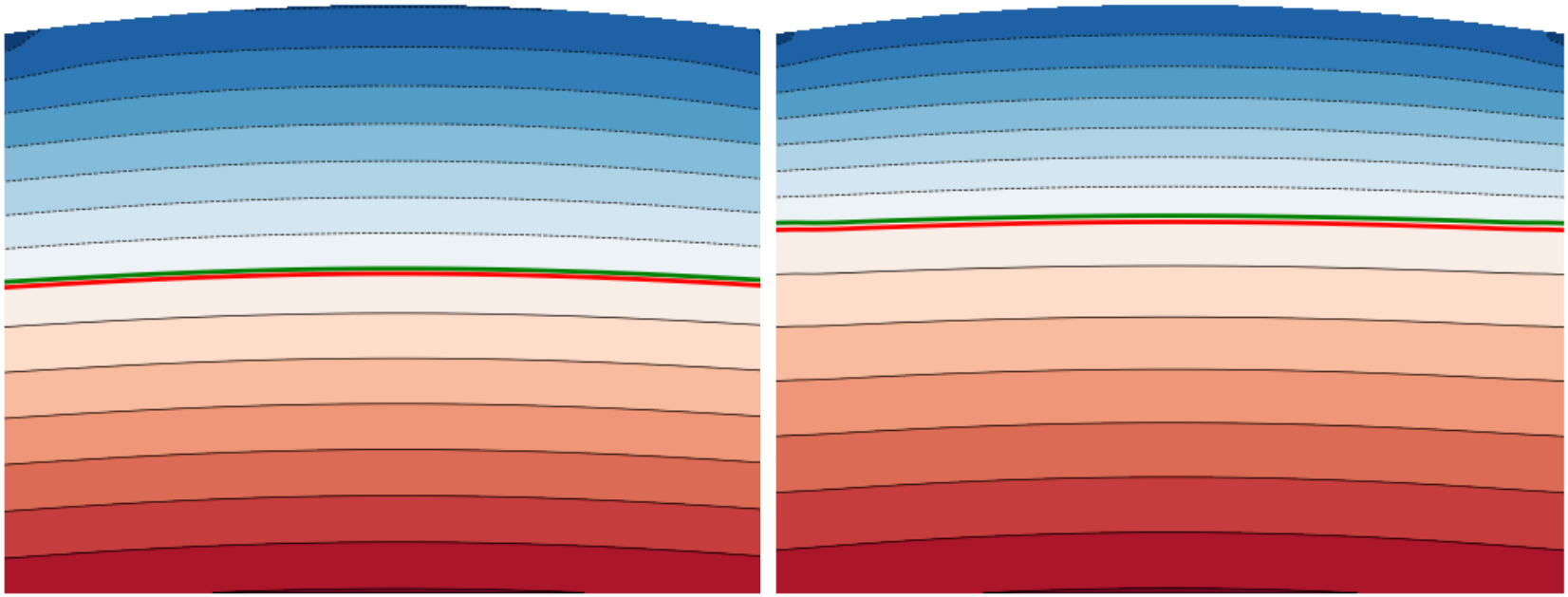


Figure: The result of the simulation procedure for the Stefan problem after 7 time steps (**left**) and 17 time steps (**right**), respectively.

Advantages and disadvantages

Advantages:

- The algorithm proposed is rather quick and uses only neighboring relations between the grid points.

- It works on non-uniform and non-rectangular grids, merging the advances of classical finite difference and finite element methods.

Disadvantages:

- The optimization procedure highly depends on the weighting, which was applied to the harmonic polynomials to obtain the optimal coefficients.

- We cannot guarantee a fixed spatial convergence (or consistency) order. For irregular geometries, even the optimal coefficients will not deliver a very accurate approximation of the differential operator.

Conclusions

- An optimization-based FD discretization of the two-dimensional Laplace operator was developed. The use and benefits of this algorithm are demonstrated in the Stefan problem.
- In this problem, the computational domains evolve with time $\Longrightarrow$ a new spatial discretization must be performed, making the speed of this process critical to the overall efficiency of the numerical method.
- The meshless, pointwise execution of this procedure provides a vectorized, more efficient alternative to the conventional assembly procedure.

Thank you for your attention!