JAX-DIPS: Differentiable Interfacial PDE Solver

Pouria A. Mistani* [†,a], Samira Pakravan[†,b], Rajesh Ilango[a], Frederic G. Gibou[b]

[a] *NVIDIA Corporation, Santa Clara, CA 95051, USA*
[b] *University of California, Santa Barbara, CA 93106-5070, USA*

Abstract

We present an end-to-end differentiable PDE solver supporting both forward and inverse free boundary problems using the level-set method. We implemented this framework using JAX, extending support for CPU/GPU/TPU platforms. Algorithmically, our proposed framework builds on the blended inverse PDE solver architecure (BiPDE) that authors have proposed earlier [3]. JAX-DIPS is an open-source software package published under MIT license and is available at https://github.com/JAX-DIPS/JAX-DIPS.

Keywords: level-set method, free boundary problems, inverse problems, jump conditions, differentiable programming

1. Introduction

Differentiable programming ...

DTO vs OTD: we are doing DTO which is Discretize then Optimize paradigm. OTD is the traditional adjoint methods.

JAX-DIPS Poisson problem solver provides the two sides of $Au = b$ which is obtained after discretization of the governing PDE problem over a uniform 3D grid. Two solution schemes for the forwards problem are possible: (1) using usual iterative methods having left-hand-side and right-hand-side of the PDE discretization, and (2) using autodifferentiation across loss function (difference between lhs and rhs in 2-norm) with respect to a given estimate for the solution vector on the underlying grid u_{ijk}. Both approaches focus on minimizing the residual over the grid points

$$\min_{u_{ijk}} ||Au - b||_2^2$$

The first method attempts to span the residual space in an iterative fashion by *estimating* the gradient of the minimizing function, while the second method offered in JAX-DIPS is directly computing the exact gradient of the minimizing function with respect to the current estimate for solution. Therefore, the advantage of this method is fewer iterations and faster convergence specially for irregular geometries where the condition number of the linear system leads to much difficulties that need to be resolved by complex preconditioning and increased number of iterations.

2. Numerical Scheme for Free Boundary Problems

Consider a closed irregular interface (Γ) that partitions the computational domain (Ω) into interior (Ω^-) and exterior (Ω^+) subdomains; *i.e.*, $\Omega = \Omega^- \cup \Gamma \cup \Omega^+$. We are interested in the

*Corresponding author: p.a.mistani@gmail.com
†These authors contributed equally to this work

solutions $\mathrm{u}^{\pm} \in \Omega^{\pm}$ to the following class of linear elliptic problems in $\mathbf{x} \in \Omega^{\pm}$:

$$\begin{aligned} &k^{\pm}u^{\pm} - \nabla \cdot (\mu^{\pm}\nabla u^{\pm}) = f^{\pm}, && \mathbf{x} \in \Omega^{\pm} \\ &[u] = \alpha, && \mathbf{x} \in \Gamma \\ &[\mu \partial_{\mathbf{n}} u] = \beta, && \mathbf{x} \in \Gamma \end{aligned}$$

Here $f^{\pm} = f(\mathbf{x} \in \Omega^{\pm})$ is the spatially varying source term, $\mu^{\pm} = \mu(\mathbf{x} \in \Omega^{\pm})$ are the diffusion coefficients, and $k^{\pm}$ are the reaction coefficients in the two domains. We consider Dirichlet boundary conditions in a cubic domain $\Omega = [-\mathrm{L}/2, \mathrm{L}/2]^3$.

2.1. The level-set method

2.2. Finite discretization method

For spatial discretizations at the presence of jump conditions we employ the numerical algorithm proposed by Bochkov and Gibou (2020) [1] (BG20) on Cartesian grids. BG20 produces second-order accurate solutions and first-order accuracte gradients in the L^{∞}-norm, while having a compact stencil that makes it a good candidate for parallelization. Moreover, treatment of the interface jump conditions do not introduce any augmented variables, this preserves the homogeneous structure of the linear system.

Here we use a finite volume discretization equation uniformly for all grid points. At grid points where the finite volumes are crossed by Γ we have

$$\sum_{s=-,+} \int_{\Omega^s \cap \mathcal{V}_{i,j}} k^s u^s d\Omega - \sum_{s=-,+} \int_{\Omega^s \cap \partial\mathcal{V}_{i,j}} \mu^s \partial_{\mathbf{n}^s} u^s d\Gamma = \sum_{s=-,+} \int_{\Omega^s \cap \mathcal{V}_{i,j}} f^s d\Omega + \int_{\Gamma \cap \mathcal{V}_{i,j}} [\mu \partial_{\mathbf{n}} u] d\Gamma$$

following standard treatment of volumetric integrals and using central differencing for derivatives we obtain in 2D (with trivial 3D extension)

$$\begin{aligned} &\sum_{s=-,+} k^s_{i,j} u^s_{i,j} |\mathcal{V}^s_{i,j}| - \sum_{s=-,+} \left(\mu^s_{i-\frac{1}{2},j} A^s_{i-\frac{1}{2},j} \frac{u^s_{i-1,j} - u^s_{i,j}}{\Delta x} + \mu^s_{i+\frac{1}{2},j} A^s_{i+\frac{1}{2},j} \frac{u^s_{i+1,j} - u^s_{i,j}}{\Delta x} + \right. \\ &\left. \mu^s_{i,j-\frac{1}{2}} A^s_{i,j-\frac{1}{2}} \frac{u^s_{i,j-1} - u^s_{i,j}}{\Delta y} + \mu^s_{i,j+\frac{1}{2}} A^s_{i,j+\frac{1}{2}} \frac{u^s_{i,j+1} - u^s_{i,j}}{\Delta y} \right) \\ &= \sum_{s=-,+} f^s_{i,j} |\mathcal{V}^s_{i,j}| + \int_{\Gamma \cap \mathcal{V}_{i,j}} \beta d\Gamma + \mathcal{O}(\max(\Delta x, \Delta y)^{\mathcal{D}}) \end{aligned}$$

where $\mathcal{D}$ is the problem dimensionality.

Note that far from interface either $s = -$ (for $\mathbf{x} \in \Omega^-$) or $s = +$ (for $\mathbf{x} \in \Omega^+$) is retained. This is automatically considered through zero values for sub-volumes $|\mathcal{V}^+_{i,j}|$ and $|\mathcal{V}^-_{i,j}|$ as well as their face areas. Note that $\mu^-_{i-1/2,j}$ (or $\mu^+_{i-1/2,j}$) corresponds to the value of diffusion coefficient at the middle of segment $A^-_{i-1/2,j}$ (or $A^+_{i-1/2,j}$) respectively, same is true for other edges as well. However, there are extra degrees of freedom on grid points whose finite volumes are crossed by the interface; *i.e.*, see double circles in figure 1. Bochkov and Gibou derived analytical expressions for the extra degrees of freedom (u^+ in Ω^- and u^- in Ω^+) in terms of the original degrees of freedom (u^- in Ω^- and u^+ in Ω^+) as well as the jump conditions, this preserves the original $\mathrm{N_x} \times \mathrm{N_y}$ system size.

In this scheme the basic idea is to extrapolate the jump at grid point from jump condition at the projected point onto the interface using a Taylor expansion: $u^+_{i,j} - u^-_{i,j} = [u]_{\mathbf{r}^{pr}_{i,j}} + \delta_{i,j}(\partial_{\mathbf{n}} u^+(\mathbf{r}^{pr}_{i,j}) - \partial_{\mathbf{n}} u^-(\mathbf{r}^{pr}_{i,j}))$. The unknown value ($u^-_{i,j}$ or $u^+_{i,j}$) is obtained based on approximation of the normal derivatives (*i.e.* $\partial_{\mathbf{n}} u^{\pm}(\mathbf{r}^{pr}_{i,j})$) which are computed using a least squares calculation on neighboring grid points that are in the fast-diffusion region (referred to as "Bias Fast") or in the slow diffusion region (referred to as "Bias Slow"). This makes two sets of ruls for unknown values $u^{\pm}_{i,j}$.

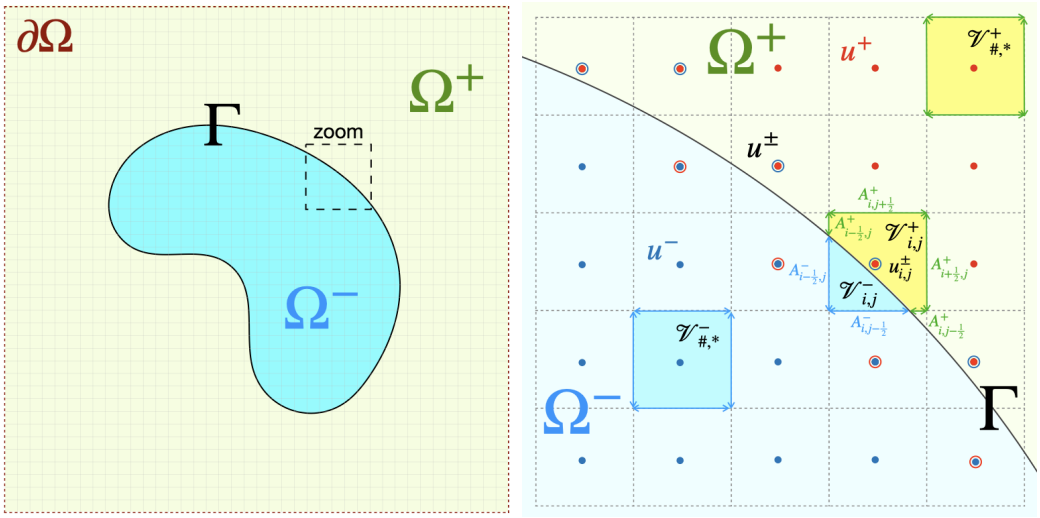

Figure 1: Notation used in this paper. Close to the interface where finite volumes are crossed by the interface, there are extra degrees of freedom (open circles) that are extrapolations of solutions from each domain to the opposite domain. Jump conditions are implicitly encoded in these extrapolated values.

In two dimensions and on uniform grids, the gradient operator at the grid cell (i, j) that is crossed by an interface is estimated by a least squares solution given by

$$(\nabla u^{\pm})_{i,j} = \mathbf{D}^{\pm}_{i,j} \begin{bmatrix} u_{i-1,j-1} - u^{\pm}_{i,j} \\ u_{i,j-1} - u^{\pm}_{i,j} \\ \vdots \\ u_{i+1,j+1} - u^{\pm}_{i,j} \end{bmatrix} \qquad \mathbf{D}^{\pm}_{i,j} = \left(X^T_{i,j} W^{\pm}_{i,j} X_{i,j}\right)^{-1} \left(W^{\pm}_{i,j} X_{i,j}\right)^T$$

and

$$W^{\pm}_{i,j} = \begin{bmatrix} \omega^{\pm}_{i,j}(-1,-1) & & & \\ & \omega^{\pm}_{i,j}(0,-1) & & \\ & & \ddots & \\ & & & \omega^{\pm}_{i,j}(1,1) \end{bmatrix} \qquad X_{i,j} = \begin{bmatrix} -h_x & -h_y \\ 0 & -h_y \\ h_x & -h_y \\ -h_x & 0 \\ 0 & 0 \\ h_x & 0 \\ -h_x & h_y \\ 0 & h_y \\ h_x & h_y \end{bmatrix}$$

and

$$\omega^{\pm}_{i,j}(p,q) = \begin{cases} 1 & (p,q) \in N^{\pm}_{i,j} \\ 0 & \text{else} \end{cases} \tag{1}$$

In this case, $D^{\pm}_{i,j}$ is a 2×9 matrix and we denote each of its 2×1 columns with $d^{\pm}_{i,j,p,q}$

$$\mathbf{D}^{\pm}_{i,j} = \begin{bmatrix} d^{\pm}_{i,j,-1,-1} & d^{\pm}_{i,j,0,-1} & d^{\pm}_{i,j,1,-1} & d^{\pm}_{i,j,-1,0} & d^{\pm}_{i,j,0,0} & d^{\pm}_{i,j,1,0} & d^{\pm}_{i,j,-1,1} & d^{\pm}_{i,j,0,1} & d^{\pm}_{i,j,1,1} \end{bmatrix}$$

The least square coefficients are then obtained by dot product of normal vector with these columns

$$c^{\pm}_{i,j,p,q} = \mathbf{n}^T_{i,j} d^{\pm}_{i,j,p,q}$$

At this point we can define a few intermediate variables at each grid point to simplify the presentation of the method,

$$\zeta^{\pm}_{i,j,p,q} := \delta_{i,j}\frac{[\mu]}{\mu^{\mp}}c^{\pm}_{i,j,p,q} \qquad \zeta^{\pm}_{i,j} := -\sum_{(p,q)\in N^{\pm}_{i,j}} \zeta^{\pm}_{i,j,p,q}$$

$$\gamma^{\pm}_{i,j,p,q} := \frac{\zeta^{\pm}_{i,j,p,q}}{1\pm\zeta^{\pm}_{i,j}} \qquad \gamma^{\pm}_{i,j} := -\sum_{(p,q)\in N^{\pm}_{i,j}} \gamma^{\pm}_{i,j,p,q}$$

where the set of neighboring grid points are

$$N^{\pm}_{i,j} = \{(p,q): \quad p=-1,0,1, \quad q=-1,0,1, \quad (p,q)\neq(0,0), \quad \mathbf{x}_{i+p,j+q}\in\Omega^{\pm}\}$$

and $\delta_{i,j}$ is the signed distance from $\mathbf{x}_{i,j}$ that is computed from the level-set function $\phi(\mathbf{x})$

$$\delta_{i,j} = \frac{\phi(\mathbf{x}_{i,j})}{|\nabla\phi(\mathbf{x}_{i,j})|}$$

- Rules based on approximating $\partial_{\mathbf{n}}u^{+}(\mathbf{r}^{pr}_{i,j})$:

$$u^{-}_{i,j} = \begin{cases} u_{i,j} & \mathbf{x}_{i,j}\in\Omega^{-} \\ u_{i,j}(1-\gamma^{-}_{i,j}) - \sum_{(p,q)\in N^{-}_{i,j}}\gamma^{-}_{i,j,p,q}u_{i+p,j+q} - (\alpha+\frac{\delta_{i,j}\beta}{\mu^{+}})(1-\gamma^{-}_{i,j}) & \mathbf{x}_{i,j}\in\Omega^{+} \end{cases} \tag{2}$$

$$u^{+}_{i,j} = \begin{cases} u_{i,j}(1-\zeta^{-}_{i,j}) - \sum_{(p,q)\in N^{-}_{i,j}}\zeta^{-}_{i,j,p,q}u_{i+p,j+q} + \alpha + \delta_{i,j}\frac{\beta}{\mu^{+}} & \mathbf{x}_{i,j}\in\Omega^{-} \\ u_{i,j} & \mathbf{x}_{i,j}\in\Omega^{+} \end{cases} \tag{3}$$

It is useful to cast this in the form of matrix operations through defining intermediate tensors:

$$\mathbf{\Gamma}_{i,j} := \begin{bmatrix} \gamma^{-}_{i-1,j+1} & \gamma^{-}_{i,j+1} & \gamma^{-}_{i+1,j+1} \\ \gamma^{-}_{i-1,j} & \gamma^{-}_{i,j} & \gamma^{-}_{i+1,j} \\ \gamma^{-}_{i-1,j-1} & \gamma^{-}_{i,j-1} & \gamma^{-}_{i+1,j-1} \end{bmatrix}, \qquad \boldsymbol{\zeta}_{i,j} := \begin{bmatrix} \zeta^{-}_{i-1,j+1} & \zeta^{-}_{i,j+1} & \zeta^{-}_{i+1,j+1} \\ \zeta^{-}_{i-1,j} & \zeta^{-}_{i,j} & \zeta^{-}_{i+1,j} \\ \zeta^{-}_{i-1,j-1} & \zeta^{-}_{i,j-1} & \zeta^{-}_{i+1,j-1} \end{bmatrix}$$

$$\mathbf{U}_{i,j} := \begin{bmatrix} u_{i-1,j+1} & u_{i,j+1} & u_{i+1,j+1} \\ u_{i-1,j} & u_{i,j} & u_{i+1,j} \\ u_{i-1,j-1} & u_{i,j-1} & u_{i+1,j-1} \end{bmatrix}, \quad \mathbf{N}^{\pm}_{i,j} := \begin{bmatrix} \omega^{\pm}_{i,j}(-1,1) & \omega^{\pm}_{i,j}(0,1) & \omega^{\pm}_{i,j}(1,1) \\ \omega^{\pm}_{i,j}(-1,0) & 0 & \omega^{\pm}_{i,j}(1,0) \\ \omega^{\pm}_{i,j}(-1,-1) & \omega^{\pm}_{i,j}(0,-1) & \omega^{\pm}_{i,j}(1,-1) \end{bmatrix}$$

where $\mathbf{N}^{-}$ is a masking filter that passes the values in the negative neighborhood of node (i,j).

We also introduce the Hadamard product $\odot$ between two identical matrices that creates another identical matrix with each entry being elementwise products. Moreover, double contraction of two tensors A and B is defined by $A:B=\sum A\odot B$ which is a scalar value and equals the sum of all entries of the Hadamard product of the tensors; *i.e.*, note $A:A$ is square of Frobenius norm of A. Using these notations, the substitution rules read

$$u^{-}_{i,j} = \begin{cases} u_{i,j} & \mathbf{x}_{i,j}\in\Omega^{-} \\ \big(1+\mathbf{\Gamma}^{-}_{i,j}:\mathbf{N}^{-}_{i,j}\big)u_{i,j} - \big(\mathbf{\Gamma}^{-}_{i,j}\odot\mathbf{N}^{-}_{i,j}\big):\mathbf{U}_{i,j} - (\alpha+\delta_{i,j}\frac{\beta}{\mu^{+}})\big(1+\mathbf{\Gamma}^{-}_{i,j}:\mathbf{N}^{-}_{i,j}\big) & \mathbf{x}_{i,j}\in\Omega^{+} \end{cases} \tag{4}$$

$$u^{+}_{i,j} = \begin{cases} \big(1+\boldsymbol{\zeta}^{-}_{i,j}:\mathbf{N}^{-}_{i,j}\big)u_{i,j} - \big(\boldsymbol{\zeta}^{-}_{i,j}\odot\mathbf{N}^{-}_{i,j}\big):\mathbf{U}_{i,j} + \alpha + \delta_{i,j}\frac{\beta}{\mu^{+}} & \mathbf{x}_{i,j}\in\Omega^{-} \\ u_{i,j} & \mathbf{x}_{i,j}\in\Omega^{+} \end{cases} \tag{5}$$

- Rules based on approximating $\partial_{\mathbf{n}}u^{-}(\mathbf{r}^{pr}_{i,j})$:

$$u^-_{i,j} = \begin{cases} u_{i,j} & \mathbf{x}_{i,j} \in \Omega^- \\ u_{i,j}(1-\zeta^+_{i,j}) - \sum_{(p,q)\in N^+_{i,j}} \zeta^+_{i,j,p,q} u_{i+p,j+q} - \alpha - \delta_{i,j}\frac{\beta}{\mu^-} & \mathbf{x}_{i,j} \in \Omega^+ \end{cases} \tag{6}$$

$$u^+_{i,j} = \begin{cases} u_{i,j}(1-\gamma^+_{i,j}) - \sum_{(p,q)\in N^+_{i,j}} \gamma^+_{i,j,p,q} u_{i+p,j+q} + (\alpha + \delta_{i,j}\frac{\beta}{\mu^-})(1-\gamma^+_{i,j}) & \mathbf{x}_{i,j} \in \Omega^- \\ u_{i,j} & \mathbf{x}_{i,j} \in \Omega^+ \end{cases} \tag{7}$$

in matrix notation we have

$$u^-_{i,j} = \begin{cases} u_{i,j} & \mathbf{x}_{i,j} \in \Omega^- \\ \big(1 + \boldsymbol{\zeta}^+_{i,j} : \mathbf{N}^+_{i,j}\big) u_{i,j} - \big(\boldsymbol{\zeta}^+_{i,j} \odot \mathbf{N}^+_{i,j}\big) : \mathbf{U}_{i,j} - \alpha - \delta_{i,j}\frac{\beta}{\mu^-} & \mathbf{x}_{i,j} \in \Omega^+ \end{cases} \tag{8}$$

$$u^+_{i,j} = \begin{cases} \big(1 + \boldsymbol{\Gamma}^+_{i,j} : \mathbf{N}^+_{i,j}\big) u_{i,j} - \big(\boldsymbol{\Gamma}^+_{i,j} \odot \mathbf{N}^+_{i,j}\big) : \mathbf{U}_{i,j} + (\alpha + \delta_{i,j}\frac{\beta}{\mu^-})\big(1 + \boldsymbol{\Gamma}^+_{i,j} : \mathbf{N}^+_{i,j}\big) & \mathbf{x}_{i,j} \in \Omega^- \\ u_{i,j} & \mathbf{x}_{i,j} \in \Omega^+ \end{cases} \tag{9}$$

2.3. 3D geometric integrations

We use uniform Cartesian grids. For computational cells that are crossed by the interface, *i.e.* $\mathcal{V}_{i,j,k} \cap \Gamma \neq 0$, we use the geometric integrations proposed by Min & Gibou (2007) [2]. In this scheme each grid cell is decomposed into five tetrahedra by the middle-cut triangulation [4] where each grid cell crossed by the interface is decomposed to five tetrahedra given by (each cell is rescaled to $[0,1]^3$):

$$\begin{aligned}
&\mathrm{T}_1 \equiv \mathrm{conv}(\mathrm{P}_{000}; \mathrm{P}_{100}; \mathrm{P}_{010}; \mathrm{P}_{001}) && \text{x = 0 face, y = 0 face, z = 0 face} \\
&\mathrm{T}_2 \equiv \mathrm{conv}(\mathrm{P}_{110}; \mathrm{P}_{100}; \mathrm{P}_{010}; \mathrm{P}_{111}) && \text{x = 1 face, y = 1 face, z = 0 face} \\
&\mathrm{T}_3 \equiv \mathrm{conv}(\mathrm{P}_{101}; \mathrm{P}_{100}; \mathrm{P}_{111}; \mathrm{P}_{001}) && \text{x = 1 face, y = 0 face, z = 1 face} \\
&\mathrm{T}_4 \equiv \mathrm{conv}(\mathrm{P}_{011}; \mathrm{P}_{111}; \mathrm{P}_{010}; \mathrm{P}_{001}) && \text{x = 0 face, y = 1 face, z = 1 face} \\
&\mathrm{T}_5 \equiv \mathrm{conv}(\mathrm{P}_{111}; \mathrm{P}_{100}; \mathrm{P}_{010}; \mathrm{P}_{001}) && \text{no face exposure}
\end{aligned}$$

2.4. Differentiable Solution Strategy

We define the loss function by the 2-norm of the residual of the discretized function:

$$\mathcal{L}(u) = ||Au - b||_2^2$$

JAX-DIPS allows for exact computation of the gradient of the loss function using automatic differentiation, *i.e.* $\nabla_u \mathcal{L}(u)$. This is advantageous over existing approximate iterative methods (with preconditioning) such as GMRES, Conjugate Gradient, *etc.* Therefore, our strategy is to leverage this capability and use more sophisticated optimizers developed in the deep learning community (*e.g.* Adam, RMSProp, *etc*) to minimize the afforementioned loss function with the desired solution vector u^* of the PDE problem.

Solving an interfacial PDE problem on a $128 \times 128 \times 128$ grid is therefore equivalent to a deep neural network architecture with $2,097,152$ trainable parameters. Besides number of trainable parameters, the computational complexity of the algebraic operations in solving a PDE system with provided discretizations closely resembles computational complexity of convolutional operations with kernel sizes equivalent to the stencil size of the discretization method (in this case $3 \times 3 \times 3$).

procedure BIAS SLOW

if $\Gamma \cap \mathcal{C}_{i,j} = \emptyset$ **then**

$B^{\pm}_{i,j} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r^{\pm}_{i,j} = 0$

else

if $\mu^{-}_{i,j} > \mu^{+}_{i,j}$ **then**

if $\phi_{i,j} \geq 0$ **then**

$B^{+}_{i,j} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r^{+}_{i,j} = 0$

$B^{-}_{i,j} = \begin{bmatrix} -\gamma^{-}_{i,j,-1,1} & -\gamma^{-}_{i,j,0,1} & -\gamma^{-}_{i,j,1,1} \\ -\gamma^{-}_{i,j,-1,0} & 1-\gamma^{-}_{i,j} & -\gamma^{-}_{i,j,1,0} \\ -\gamma^{-}_{i,j,-1,-1} & -\gamma^{-}_{i,j,0,-1} & -\gamma^{-}_{i,j,1,-1} \end{bmatrix}$; $r^{-}_{i,j} = -(\alpha^{proj}_{i,j} + \delta_{i,j}\frac{\beta^{proj}_{i,j}}{\mu^{+}_{proj}})(1-\gamma^{-}_{i,j})$

else

$B^{+}_{i,j} = \begin{bmatrix} -\zeta^{-}_{i,j,-1,1} & -\zeta^{-}_{i,j,0,1} & -\zeta^{-}_{i,j,1,1} \\ -\zeta^{-}_{i,j,-1,0} & 1-\zeta^{-}_{i,j} & -\zeta^{-}_{i,j,1,0} \\ -\zeta^{-}_{i,j,-1,-1} & -\zeta^{-}_{i,j,0,-1} & -\zeta^{-}_{i,j,1,-1} \end{bmatrix}$; $r^{+}_{i,j} = \alpha^{proj}_{i,j} + \delta_{i,j}\frac{\beta^{proj}_{i,j}}{\mu^{+}_{proj}}$

$B^{-}_{i,j} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r^{-}_{i,j} = 0$

else

if $\phi_{i,j} \geq 0$ **then**

$B^{+}_{i,j} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r^{+}_{i,j} = 0$

$B^{-}_{i,j} = \begin{bmatrix} -\zeta^{+}_{i,j,-1,1} & -\zeta^{+}_{i,j,0,1} & -\zeta^{+}_{i,j,1,1} \\ -\zeta^{+}_{i,j,-1,0} & 1-\zeta^{+}_{i,j} & -\zeta^{+}_{i,j,1,0} \\ -\zeta^{+}_{i,j,-1,-1} & -\zeta^{+}_{i,j,0,-1} & -\zeta^{+}_{i,j,1,-1} \end{bmatrix}$; $r^{-}_{i,j} = \alpha^{proj}_{i,j} + \delta_{i,j}\frac{\beta^{proj}_{i,j}}{\mu^{-}_{proj}}$

else

$B^{+}_{i,j} = \begin{bmatrix} -\gamma^{+}_{i,j,-1,1} & -\gamma^{+}_{i,j,0,1} & -\gamma^{+}_{i,j,1,1} \\ -\gamma^{+}_{i,j,-1,0} & 1-\gamma^{+}_{i,j} & -\gamma^{+}_{i,j,1,0} \\ -\gamma^{+}_{i,j,-1,-1} & -\gamma^{+}_{i,j,0,-1} & -\gamma^{+}_{i,j,1,-1} \end{bmatrix}$; $r^{+}_{i,j} = (\alpha^{proj}_{i,j} + \delta_{i,j}\frac{\beta^{proj}_{i,j}}{\mu^{-}_{proj}})(1-\gamma^{+}_{i,j})$

$B^{-}_{i,j} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r^{-}_{i,j} = 0$

Algorithm 1: Bias Slow approximation of the non-existing solution value on a grid point based on existing solution values in its neighborhood. The notation is used for $u^{\pm}_{i,j} = B^{\pm}_{i,j} : \mathbf{U}_{i,j} + r^{\pm}_{i,j}$.

3. Numerical Examples

$$
\begin{aligned}
k^{\pm}u^{\pm} - \nabla \cdot (\mu^{\pm}\nabla u^{\pm}) &= f^{\pm}, && \mathbf{x} \in \Omega^{\pm} \\
[u] &= \alpha, && \mathbf{x} \in \Gamma \\
[\mu \partial_{\mathbf{n}} u] &= \beta, && \mathbf{x} \in \Gamma
\end{aligned}
$$

We present successively more complex test cases and analyze performance of `JAX-DIPS` in each case.

3.1. No interface, $\Gamma = \emptyset$

We set the level-set function to $\phi(\mathbf{x}) = \sqrt{x^2 + y^2 + z^2} + 0.5$ within a domain $\Omega : [-1, 1]^3$ characterizing absence of all jump conditions. Using the method of manufactured solutions, we construct the following Poisson problem for an exact solution $u(\mathbf{x}) = \sin(y)\cos(x)$ with appropriate Dirichlet boundary conditions:

$$
\begin{aligned}
-\Delta u &= 2\sin(y)\cos(x), && \mathbf{x} \in \Omega \\
u(\mathbf{x}) &= \sin(y)\cos(x), && \mathbf{x} \in \partial\Omega
\end{aligned}
$$

with `Adam` optimizer starting from an initial condition $\hat{u}(\mathbf{x}; t = 0) = y$ which does not satisfy the system of equations.

Acknowledgement

References

[1] D. Bochkov and F. Gibou. Solving elliptic interface problems with jump conditions on cartesian grids. *Journal of Computational Physics*, 407:109269, 2020.

[2] C. Min and F. Gibou. Geometric integration over irregular domains with application to level-set methods. *Journal of Computational Physics*, 226(2):1432–1443, 2007.

[3] S. Pakravan, P. A. Mistani, M. A. Aragon-Calvo, and F. Gibou. Solving inverse-pde problems with physics-aware neural networks. *Journal of Computational Physics*, 440:110414, 2021.

[4] J. F. Sallee. The middle-cut triangulations of the n-cube. *SIAM Journal on Algebraic Discrete Methods*, 5(3):407–419, 1984.