JAX-DIPS: Differentiable Interfacial PDE Solver

Pouria A. Mistani* †,a, Samira Pakravan†,b, Rajesh Ilangoa, Frederic G. Giboub

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 [4]. 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 $u^\pm \in \Omega^\pm$ to the following class of linear elliptic problems in $\mathbf{x} \in \Omega^\pm$:

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

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)

$$\sum_{s=-,+} k_{i,j}^s u_{i,j}^s |\mathcal{V}_{i,j}^s| - \sum_{s=-,+} \left(\mu_{i-\frac{1}{2},j}^s A_{i-\frac{1}{2},j}^s \frac{u_{i-1,j}^s - u_{i,j}^s}{\Delta x} + \mu_{i+\frac{1}{2},j}^s A_{i+\frac{1}{2},j}^s \frac{u_{i+1,j}^s - u_{i,j}^s}{\Delta x} + \right.$$
$$\left. \mu_{i,j-\frac{1}{2}}^s A_{i,j-\frac{1}{2}}^s \frac{u_{i,j-1}^s - u_{i,j}^s}{\Delta y} + \mu_{i,j+\frac{1}{2}}^s A_{i,j+\frac{1}{2}}^s \frac{u_{i,j+1}^s - u_{i,j}^s}{\Delta y} \right)$$
$$= \sum_{s=-,+} f_{i,j}^s |\mathcal{V}_{i,j}^s| + \int_{\Gamma \cap \mathcal{V}_{i,j}} \beta d\Gamma + \mathcal{O}(\max(\Delta x, \Delta y)^{\mathcal{D}})$$

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: $u_{i,j}^+ - u_{i,j}^- = [u]_{\mathbf{r}_{i,j}^{pr}} + \delta_{i,j}(\partial_\mathbf{n} u^+(\mathbf{r}_{i,j}^{pr}) - \partial_\mathbf{n} u^-(\mathbf{r}_{i,j}^{pr}))$. The unknown value ($u_{i,j}^-$ or $u_{i,j}^+$) is obtained based on approximation of the normal derivatives (*i.e.* $\partial_\mathbf{n} u^\pm(\mathbf{r}_{i,j}^{pr})$) 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 $u_{i,j}^\pm$.
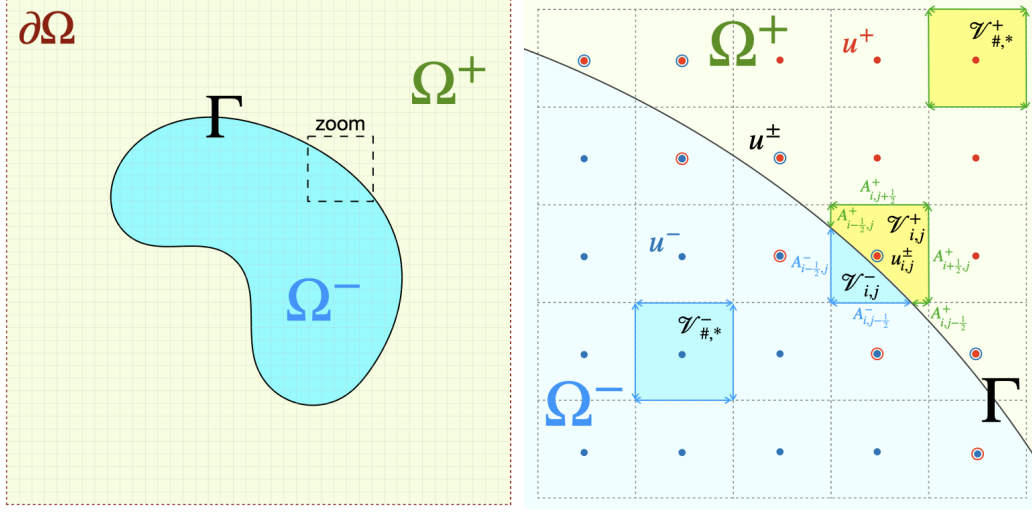
Figure 1: Notation used in this paper. Close to the interface where finite volumes are crossed by the interface, there are extra degrees of freedom (open circles) that are extrapolations of solutions from each domain to the opposite domain. Jump conditions are implicitly encoded in these extrapolated 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

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

and

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

and

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

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

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

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

$$c_{i,j,p,q}^{\pm} = \mathbf{n}_{i,j}^{T} d_{i,j,p,q}^{\pm}$$

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

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

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

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 $\partial_{\mathbf{n}}u^{+}(\mathbf{r}^{pr}_{i,j})$:

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

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

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

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

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

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

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

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

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

4

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

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

in matrix notation we have

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

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

2.3. 3D surface and volume 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) [3]. In this scheme each grid cell, $\mathcal{C}$, is decomposed into five tetrahedra by the middle-cut triangulation [5] (each cell is rescaled to $[0,1]^3$) that are described below (also see figure 1 (right) of [3]):

$$S_1 \equiv \text{conv}(P_{000}; P_{100}; P_{010}; P_{001}) \qquad \text{x = 0 face, y = 0 face, z = 0 face}$$
$$S_2 \equiv \text{conv}(P_{110}; P_{100}; P_{010}; P_{111}) \qquad \text{x = 1 face, y = 1 face, z = 0 face}$$
$$S_3 \equiv \text{conv}(P_{101}; P_{100}; P_{111}; P_{001}) \qquad \text{x = 1 face, y = 0 face, z = 1 face}$$
$$S_4 \equiv \text{conv}(P_{011}; P_{111}; P_{010}; P_{001}) \qquad \text{x = 0 face, y = 1 face, z = 1 face}$$
$$S_5 \equiv \text{conv}(P_{111}; P_{100}; P_{010}; P_{001}) \qquad \text{no face exposure}$$

Hence each 3D grid cell is the union of 5 tetrahedra (simplices) $\mathcal{C} = \cup_{i=1}^5 S_i$, where each simplex is identified by the pre-existing vertices of the grid cell (hence not creating new grid points). Given the values of the level set function sampled at these vertices one can compute coordinates of intersection points of the interface with each of the simplices $S_i \cap \Gamma$ as well as the negative domain $S_i \cap \Omega^-$. If $P_0, \cdots, P_3$ are the four vertices of a simplex S, then Γ crosses an edge $P_i P_j$ if and only if $\phi(P_i)\phi(P_j) < 0$ and the intersection point across this edge is given by linear interpolation:

$$P_{ij} = P_j \frac{\phi(P_i)}{\phi(P_i) - \phi(P_j)} - P_i \frac{\phi(P_j)}{\phi(P_i) - \phi(P_j)}$$

Number of negative level-set values on the 4 (in 3D) tetrahedron vertices classifies the specific configuration for intersection between simplex S and the interface through a variable $\eta(\phi, S) = n(P_i|\phi(P_i) < 0)$. In 3D, possible values are $\eta = 0, 1, 2, 3, 4$ that correspond to the four configurations for the intersection cross section enumerated below:

- $S \cap \Gamma$, see table 2 and figure 2 of [3]:

  - $\eta = 0$: tetrahedron (S) is completely in positive domain with no intersection, $S \cap \Gamma = \emptyset$.

  - $\eta = 1$: with a single vertex in negative domain and remaining three in positive domain, the tetrahedron and interface have exactly 3 intersection points, the simplex $S \cap \Gamma$ has exactly 3 vertices; *cf.*, see figure 2 (center) of [3].

  - $\eta = 2$: with two vertices in negative domain and remaining two in positive domain, the cross section has four vertices that is splitted into two 3-vertex simplices; *cf.*, see figure 2 (right) of [3].

5

1: procedure BIAS SLOW

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

3: $B_{i,j}^{\pm} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r_{i,j}^{\pm} = 0$

4:   **else**

5:   **if** $\mu_{i,j}^{-} > \mu_{i,j}^{+}$ **then**

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

7: $B_{i,j}^{+} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r_{i,j}^{+} = 0$

8: $B_{i,j}^{-} = \begin{bmatrix} -\gamma_{i,j,-1,1}^{-} & -\gamma_{i,j,0,1}^{-} & -\gamma_{i,j,1,1}^{-} \\ -\gamma_{i,j,-1,0}^{-} & 1-\gamma_{i,j}^{-} & -\gamma_{i,j,1,0}^{-} \\ -\gamma_{i,j,-1,-1}^{-} & -\gamma_{i,j,0,-1}^{-} & -\gamma_{i,j,1,-1}^{-} \end{bmatrix}$; $r_{i,j}^{-} = -(\alpha_{i,j}^{proj} + \delta_{i,j}\frac{\beta_{i,j}^{proj}}{\mu_{proj}^{+}})(1-\gamma_{i,j}^{-})$

9:   **else**

10: $B_{i,j}^{+} = \begin{bmatrix} -\zeta_{i,j,-1,1}^{-} & -\zeta_{i,j,0,1}^{-} & -\zeta_{i,j,1,1}^{-} \\ -\zeta_{i,j,-1,0}^{-} & 1-\zeta_{i,j}^{-} & -\zeta_{i,j,1,0}^{-} \\ -\zeta_{i,j,-1,-1}^{-} & -\zeta_{i,j,0,-1}^{-} & -\zeta_{i,j,1,-1}^{-} \end{bmatrix}$; $r_{i,j}^{+} = \alpha_{i,j}^{proj} + \delta_{i,j}\frac{\beta_{i,j}^{proj}}{\mu_{proj}^{+}}$

11: $B_{i,j}^{-} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r_{i,j}^{-} = 0$

12:   **else**

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

14: $B_{i,j}^{+} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r_{i,j}^{+} = 0$

15: $B_{i,j}^{-} = \begin{bmatrix} -\zeta_{i,j,-1,1}^{+} & -\zeta_{i,j,0,1}^{+} & -\zeta_{i,j,1,1}^{+} \\ -\zeta_{i,j,-1,0}^{+} & 1-\zeta_{i,j}^{+} & -\zeta_{i,j,1,0}^{+} \\ -\zeta_{i,j,-1,-1}^{+} & -\zeta_{i,j,0,-1}^{+} & -\zeta_{i,j,1,-1}^{+} \end{bmatrix}$; $r_{i,j}^{-} = \alpha_{i,j}^{proj} + \delta_{i,j}\frac{\beta_{i,j}^{proj}}{\mu_{proj}^{-}}$

16:   **else**

17: $B_{i,j}^{+} = \begin{bmatrix} -\gamma_{i,j,-1,1}^{+} & -\gamma_{i,j,0,1}^{+} & -\gamma_{i,j,1,1}^{+} \\ -\gamma_{i,j,-1,0}^{+} & 1-\gamma_{i,j}^{+} & -\gamma_{i,j,1,0}^{+} \\ -\gamma_{i,j,-1,-1}^{+} & -\gamma_{i,j,0,-1}^{+} & -\gamma_{i,j,1,-1}^{+} \end{bmatrix}$; $r_{i,j}^{+} = (\alpha_{i,j}^{proj} + \delta_{i,j}\frac{\beta_{i,j}^{proj}}{\mu_{proj}^{-}})(1-\gamma_{i,j}^{+})$

18: $B_{i,j}^{-} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$; $r_{i,j}^{-} = 0$

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 $u_{i,j}^{\pm} = B_{i,j}^{\pm} : \mathbf{U}_{i,j} + r_{i,j}^{\pm}$.

- $\eta = 3$: with one vertex in positive domain and remaining three vertices in negative domain, the cross section has 3 vertices that is computed by inverting the sign of the level-set values on vertices and following the instruction for case $\eta = 1$.

- $\eta = 4$: tetrahedron is completely in negative domain with no intersection, $S \cap \Gamma = \emptyset$.

- $S \cap \Omega^-$, see table 4 and figure 4 of [3]:

  - $\eta = 0$: tetrahedron is completely in positive domain with no intersection, $S \cap \Omega^- = \emptyset$;

  - $\eta = 1$: the intersection $S \cap \Omega^-$ is characterized by a single tetrahedron with 4 vertices according to figure 4 (left) of [3]; *i.e.*, one vertex is the negative level-set vertex of the parent tetrahedron and three others are interpolated points over the three edges pertaining to the negative vertex.

  - $\eta = 2$: the intersection $S \cap \Omega^-$ is characterized by three tetrahedra with 12 vertices according to figure 4 (center) of [3]. Note that there

  - $\eta = 3$: the intersection $S \cap \Omega^-$ is characterized by three tetrahedra with 12 vertices according to figure 4 (right) of [3].

  - $\eta = 4$: tetrahedron is completely in negative domain and $S \cap \Omega^- = S$;

Note that although we only need to allocate memory for at most 4 vertices to uniquely identify $S \cap \Gamma$, in JAX-DIPS we choose to pre-allocate memory for two 3-vertex simplex data structure per S with a total of 6 vertices to separately store information for the cross section geometry. Similarly for $S \cap \Omega^-$ we pre-allocate memory for a three 4-vertex simplex data structure per S. Altogether, in the current implementation the geometric information of intersection points for each simplex S is expressed in terms of 5 simplices (2 three-vertex simplices for surface area and 3 four-vertex simplices for volume) using 18 points; this is an area for future optimization.

Having the intersection points, we compute surface and volume integrals of a given field over the interface Γ and in negative domain Ω^- as a summation of integrals over the identified simplices. For each simplex (with $n = 3$ or $n = 4$ vertices) surface and volume integrals can be numerically computed by having these vertices $P_0, \cdots, P_n$ and the values of the field f at these vertices according to

$$\int_S f dx = \text{vol}(S) \cdot \frac{f(P_0) + \cdots + f(P_n)}{n + 1}$$

where

$$\text{vol}(S) = \frac{1}{n!} \left| \det \begin{pmatrix} (P_1 - P_0)\hat{e}_1 & \cdots & (P_n - P_0)\hat{e}_1 \\ \vdots & \ddots & \vdots \\ (P_1 - P_0)\hat{e}_n & \cdots & (P_n - P_0)\hat{e}_n \end{pmatrix} \right|$$

with $\hat{e}_i$ being the ith Cartesian unit basis vector.

2.4. Cross sections of interface with grid cell faces

For the numerical discretizations considered in this work we also need the surface areas for simplices created at the intersection of Γ with each of the 6 faces of a grid cell $\mathcal{C}$. In JAX-DIPS for each face we reuse two corresponding simplices exposed to that face that were calculated in the geometric integrations module, explicitly:

- $x = 0$ face has contributions from (S_1, S_4)

- $x = 1$ face has contributions from (S_2, S_3)

- $y = 0$ face has contributions from (S_1, S_3)

- $y = 1$ face has contributions from (S_2, S_4)

- $z = 0$ face has contributions from (S_1, S_2)

- $z = 1$ face has contributions from (S_3, S_4)

For each face, we extract vertices from (S_i, S_j)-pair that lie on the considered face and sum the surface areas in (negative domain) contributed from each simplex on that face; therefore, the portion of the face surface area in the positive domain is simply the complementing value area$^+$ = area$_{\text{face}}$ − area$^-$; *i.e.*, this ensures sum of areas adds up to the exact face area in downstream computations.

2.5. 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 $3 \times 3 \times 3$).

3. Numerical Examples

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

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:

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

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

3.2. Interface, $\Gamma \neq \emptyset$

We consider the example 4.6 of the Voronoi-Interface Method (VIM) of Guittet et al 2015 [2] where a sphere $\phi(\mathbf{x}) = \sqrt{x^2 + y^2 + z^2} - 0.5$ is centered in a box $\Omega : [-1, 1]^3$ with the exact solution

$$
\begin{aligned}
u^-(x, y, z) &= e^z, & \phi(\mathbf{x}) < 0 \\
u^+(x, y, z) &= \cos(x)\sin(y), & \phi(\mathbf{x}) \geq 0
\end{aligned}
$$

and the diffusion coefficient

$$
\begin{aligned}
\mu^-(x, y, z) &= y^2 \ln(x + 2) + 4 & \phi(\mathbf{x}) < 0 \\
\mu^+(x, y, z) &= e^{-z} & \phi(\mathbf{x}) \geq 0
\end{aligned}
$$

that imply

$$
\begin{aligned}
f^-(x, y, z) &= -[y^2 \ln(x + 2) + 4]e^z & \phi(\mathbf{x}) < 0 \\
f^+(x, y, z) &= 2\cos(x)\sin(y)e^{-z} & \phi(\mathbf{x}) \geq 0
\end{aligned}
$$

Table 8 of [2] reports convergence results for the solution and its gradient over the surface of the sphere in the L^∞-norm, here we report similar results for comparison with VIM.

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] A. Guittet, M. Lepilliez, S. Tanguy, and F. Gibou. Solving elliptic problems with discontinuities on irregular domains–the voronoi interface method. *Journal of Computational Physics*, 298:747–765, 2015.

[3] 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.

[4] 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.

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