1 Algorithm DevOps: HRBF-LMAPS

1.1 Grid Initialization

Goal: Generate computational nodes (interior and boundary) in a 2D domain.

Mathematics:

$$\text{Domain: } \Omega = [0, 1] \times [0, 1] = [X \quad Y]$$
$$[x \quad y \quad \vec{0}] = \text{reshape}(X/Y \quad [] \quad 1)$$

Nodes are distributed as:

- **Boundary nodes:** $x = 0$ or $x = 1$, $y = 0$ or $y = 1$.
- **Interior nodes:** $(x, y) \in \Omega \setminus \partial\Omega$.

Implementation:

- Input number of subdivisions/resolution, distribution case, other constant parameters.
- Select case of implementation, depending on the shape [rectangular, circular, Sobol sequential] of the stencil.
- Sort ($0 :=$ interior, $\mathbb{Z}^+ :=$ boundary). Returns coordinates as output 'coor'.
- Flatten grid and initialize coordinates as 'coor = [x y 0].' Classify nodes, assign boundary types.
- Collocate in a single list, plot the domain stencil.

1.2 Boundary Condition Setup

Goal: Specify velocity and pressure conditions on the domain boundaries.

Mathematics:

- Velocity (u, v):
$$u = 1,\ v = 0 \quad \text{on inflow } (y = 0), \quad u = v = 0 \quad \text{elsewhere.}$$
- Pressure (p):
$$p = 0 \quad \text{(or derived using velocity gradients at boundaries).}$$

Implementation: Initialize velocity fields u,v as zero everywhere, then impose boundary conditions.

1.3 Localized Gradients and Weights

Goal: Use RBFs and Hermite interpolation to compute localized approximations of gradients and Laplacian and retrieve weights.

Mathematics:

- Define radial basis function $\phi(\vec{r})$, derivative terms, where $\vec{r} = \text{dist}(r_i, r_j)$

$$\begin{aligned}
\phi(c, r) &= \sqrt{1 + r^2c^2}\left(\frac{r^2}{9} + \frac{4}{9c^2}\right) - \frac{\log(\sqrt{1 + c^2r^2} + 1)}{3c^2} \\
d_1\phi(c, r) &= -\sqrt{1 + (rc)^2} \quad \text{(1st derivative)} \\
d_2\phi(c, r) &= -\sqrt{1 + (rc)^2} \quad \text{(2nd derivative)} \\
d_{12}\phi(c, r) &= \frac{2c^2}{\sqrt{1 + (rc)^2}} - \frac{c^4r^2}{(1 + (rc)^2)^{3/2}} \quad \text{(Mixed derivative)}
\end{aligned}$$

- Solve the augmented system to compute weights: $A_H\vec{w} = \vec{b} \implies \vec{w} = A_H^{-1}\vec{b}$.

$$A_H = \begin{bmatrix} \phi(c,\ \mathrm{r}) & d_2\phi & 1 \\ d_1\phi & d_{12}\phi & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Here, the RHS is given as follows:

$$\vec{b} = \begin{bmatrix} \vec{f} \\ \mathcal{L}\vec{f} \\ \vec{0} \end{bmatrix} = \begin{bmatrix} \mathcal{L}_1\phi(\vec{r})^n \\ \mathcal{L}_1\mathcal{L}_2\phi(\vec{r})^m \\ \vec{0} \end{bmatrix} = \begin{bmatrix} d_2\phi \\ d_{12}\phi \\ 0 \end{bmatrix}$$

Implementation:

- Initialize the global sparse matrix system by constructing a KD-tree with traversal for efficient nearest-neighbors search.
- (Reorientation of each local system into a local coordinate system.)
- Define forcing term $f(\vec{x}) = \mathcal{L}u(\vec{x})$, and we know u_1, v_1 from initial conditions, allowing us to solve the momentum equation (next step).
- Retrieve and store weights.

1.4 Solve Momentum and Poisson Equations

Goal: Update velocity and pressure gradient fields using the momentum and Poisson-pressure equations.

Mathematics:

- Define forcing terms as the radial basis function, boundary terms, leading to the velocity terms being:

$$\hat{u}(\vec{r}) = \sum_{j=1}^{n} a_j\phi(\|r - r_j\|, c) + \sum_{l=1}^{q} \alpha_{n+l}P_l(\|r - r_j\|)\ ; \quad \sum_{i=1}^{n} \alpha_i P_l(\vec{p}_i) = 0$$

- Use a compact stencil with Hermite derivatives:

$$\nabla u = \mathcal{L}u = \sum_{j=1}^{n} w_k^{(j)}u_k^{(j)} + \sum_{j=1}^{m} \tilde{w}_k^{(j)}\mathcal{L}u_k^{(j)}\ ; \quad \mathcal{L}u(\vec{r}) := f(\vec{r})$$

- Solve the Poisson equation with Dirichlet boundary conditions:

$$\nabla p = -\nabla^2 u = f(x, y), \quad u = g(x, y) \text{ on the boundary.}$$

$$\mathbf{p} = A_p^{-1}\ \mathbf{rhs}$$

- Time discretization (Euler):

$$u^{n+1} = u^n + \Delta t\left[\nu\nabla^2 u^n - (u^n\nabla u^n) - \nabla p^n\right].$$

Implementation:

- Define the forcing, boundary terms and polynomials ($\vec{1}$).
- Call ‘knnsearch’ function to form local stencil of nearest neighborhood.
- Solve the modified momentum equation, using weights to compute gradient and Laplacian terms.
- Solve the Poisson equation, by assembling the global sparse matrix for $\nabla^2 p$ using local RBF weights and intermediate u, v values.

1.5 Enforce Boundary Condition

Goal: Ensure updated velocity and pressure satisfy physical constraints.

Mathematics:

- Set $u = 0, v = 0$ at solid boundaries.
- For free-slip or inflow boundaries: - Velocity or pressure gradient imposed based on problem setup.

Implementation:

- Directly modify boundary values in arrays u, v, p.

1.6 Visualization and Benchmarking

Goal: Display velocity field, pressure distribution, and convergence over time.

Implementation:

- Use `quiver` to plot velocity vectors.
- Use `scatter` for pressure distribution.
- Streamline visualization for flow trajectories.