AI4PDE

FEM,Mesh and Sampling

Yunfeng Liao

26 July,2024

Presentation Overview

Section Overview

Sources

Literatures

- The Deep Ritz method: A deep learning-based numerical algorithm for solving variational problems.Weinan E and Bing Yu.Communications in Mathematics and Statistics,2017.
- hp-VPINNs. EHSAN KHARAZMI et al. **Computer Methods in Applied Mechanics and Engineering,2021**.
- MeshingNet: A new mesh generation method based on deep learning.Zheyan Zhang et al. **ICCS, 2022.**
- M2N: Mesh movement networks for PDE solvers. Wenbin Song et al. **NIPS, 2022.**
- Efficient Training of Physics-Informed Neural Networks via Importance Sampling. MA Nabian. Computer-Aided Civil and Infrastructure Engineering, 2021.
- DMIS: Dynamic Mesh-based Importance Sampling for Training Physics-Informed Neural Networks. Zijiang Yang et al. **AAAI,2023**.
- GAS: A Gaussian Mixture Distribution-Based Adaptive Sampling Method for PINNs. **EAAI,2024**.

Tutorials

- Partial Differential Equations. Evans.
- 微分方程数值解法. 李荣华,刘播.
- 数值分析. 李庆扬等.
- Finite Element Methods and Their Applications. Zhangxin Chen.
- Introductory Variational Calculus on Manifolds. Ivo Terek.
- From Fourier Analysis to Wavelets. Jonas Gomes, Luiz Velho.

Section Overview

Retrospect

Ways to Solve PDE

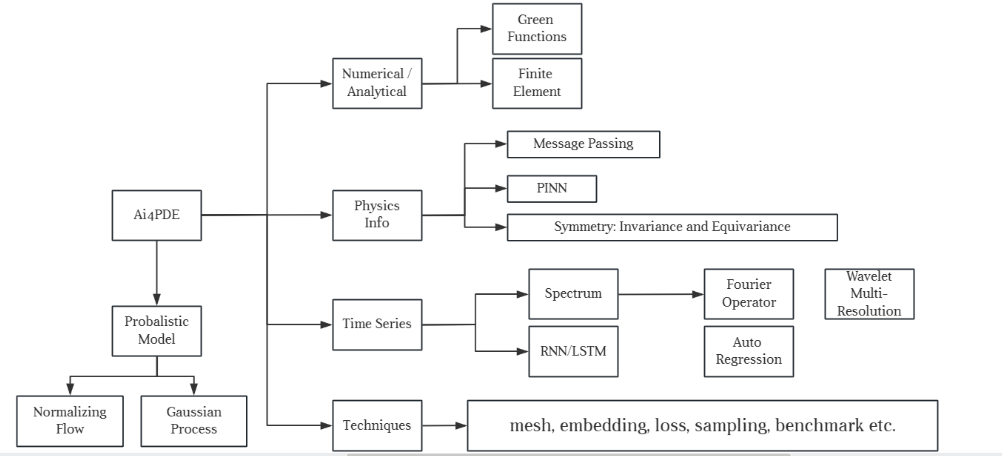


Fig. 1: Ways to Solve PDE with DL

Section Overview

FEM

FEM:Finite Element Method

Example

Consider a stationary problem in the **strong form**:

$$\Delta u(x) = -f, x \in \Omega \tag{1}$$

$$u(x) = 0, x \in \partial\Omega \tag{2}$$

PINN handles it directly, but it also has a **weak form** given by the inner product over $\Omega, \partial\Omega$. By Green's first identity:

$$(\Delta u, v)_\Omega = -(\nabla u, \nabla v)_\Omega = (-f, v)_\Omega, \forall v \in H_0^1(\mathbb{R}) \tag{3}$$

One can also use (3) to construct the loss terms. In the sequel, we define a bilinear form:

$$a(u, v) := (\nabla u, \nabla v)_\Omega \tag{4}$$

FEM

Theorem

One can show that, to look for u minimizing $F(u)$ in (5) is **equivalent** to solve the weak form (3).

$$F(u) := \frac{1}{2}a(u,u) - (f,u) \tag{5}$$

Remark

It is easy to show that the u's obtained from the strong and weak forms only differs up to a zero measure set.

Thus, instead of dealing with the optimization problem directly as **the deep Ritz method**, FEM decomposes u into the basis functions and solves **the linear equations derived from the weak form** on a **certain grid**.

FEM

On a triangulation mesh, for instance, ϕ_j is the basis function s.t. $\phi_j(\mathbf{x}_i) = \delta_{ij}$ where $\mathbf{x}_i$ is the ith vertex. It gives the linear equations:

$$u := \sum_i u_i \phi_i \tag{6}$$

$$\mathbf{A}_{ij} := a(\phi_i, \phi_j), \mathbf{L}_i := (f, \phi_i) \tag{7}$$

$$\mathbf{A}\mathbf{u} = \mathbf{L} \tag{8}$$

Terminology

The error that u can't be exactly expressed via ϕ_i is termed **approximation gap** and the error from the ground truth within span$\{\phi_i\}$ is termed **amortization gap**.

Remark

$u|_K$ is usually a **polynomial** defined in a finite element triangle,rectangle and prism Ketc, which is a **polynomial determined by values on some selected points**.

FEM

The $u|_K \in Q_r(K)$ where:

$$Q_r(K) := \{v : v(x) = \sum_{i,j}^{r} v_{ij} x^i y^j\} \tag{9}$$

Fig. 1.17. The element degrees of freedom for $Q_1(K)$

Fig. 1.18. The element degrees of freedom for $Q_2(K)$

Fig. 2: Examples of Finite Elements

Remark

We can always find a weak form and an equivalent variation form for any elliptic PDE with any boundary condition(even discontinuous). But this doesn't hold water for arbitrary PDE.

How to Introduce DL?

Potential Problems and Possible incorporation can be:

- It may be tough to work out lots of integrals $(\phi_i, \phi_j), (f, \phi_i)$ although some quadratures come into effect.
- The analysis of a high dimension finite element can be rather complicated.
- A high-quality mesh is need before applying FEM. It is hard to tell which sort of partition is best.
- ...

Deep Ritz Method [1]

Trivial. The value of the functional to minimize is nothing but the loss.

$$\mathscr{L} := F(\mathrm{NN}(\mathbf{x})) := \frac{1}{2} a(\mathrm{NN}(\mathbf{x}), \mathrm{NN}(\mathbf{x})) - (f, \mathrm{NN}(\mathbf{x})) \tag{10}$$

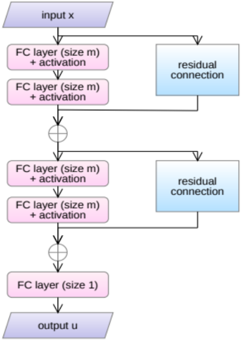


Fig. 3: Deep Ritz Method

H^p-VPINN [2]

Also trivial. Enlightened by the concept, **test functions** in the variation, the loss is given by a set of functions $v \in V_k$. Since $(\Delta u, v)_\Omega = (-f, v)_\Omega, \forall v \in L^2(\mathbb{R})$,

$$\mathscr{L} := \text{Average}_k \, \text{abs}((\text{NN}(\mathbf{x}), v_k) + (f, v_k)) + k_1 \, \text{BoundaryLoss} + k_2 \, \text{DataLoss} \tag{11}$$

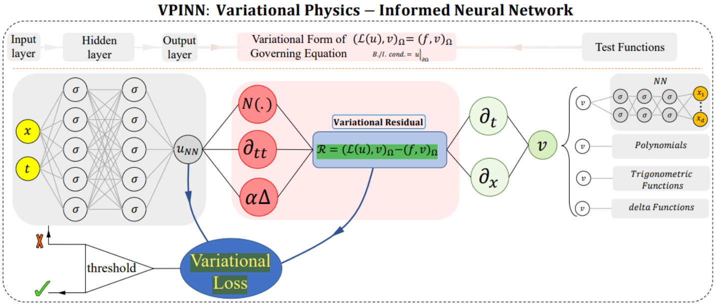


Fig. 4: VPINN

Section Overview

Mesh

A High-Quality Mesh Matters in FEM

Spirit

The more sensitive at a certain location, the denser the mesh is.

We wish to have a mesh generator for a type of PDEs.

Challenges

- How to **represent and embed** info like boundary geometry, PDE parameters, boundary conditions and coordinates efficiently?
- What are the crucial components to evaluate by NN **when generating the mesh**?
- Can the mesh have some **invariant and equivariant properties**?
- Can the generator **adaptively** gives a mesh w.r.t. the current system states?

Demonstration

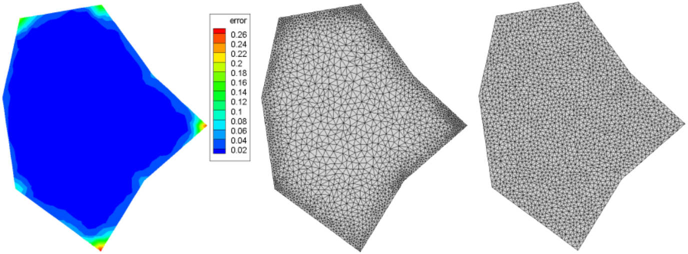
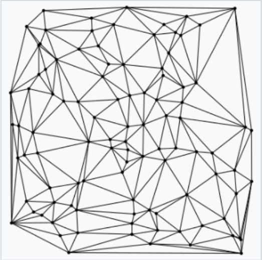


Fig. 5: End Product

Preliminary: Delaunay Triangulation

Given n points on a plane, one can yield a Delaunay Triangulation within time complexity $O(n \log n)$.
The triangulation behaves well, that is, there is no other points within the circumcircle of any triangle.



A Delaunay triangulation of a random set of 100 points in a plane.

Fig. 6: Delaunay Triangulation

Preliminary: Mean Value Coordinates

A naive equivariant coordinate representation can be a convex linear combination of the vertices:

$$\sum_{i=1}^{k} \lambda_i \mathbf{v}_i = \mathbf{v}_0, \sum_{i=1}^{k} \lambda_i = 1, \lambda_i \geq 0 \tag{12}$$

But unfortunately, the solution of λ is a bounded manifold, not even a space. The coordinate is expected to be smooth,unique and equivariant. In light of the mean theorem in harmonic field, Michael proposed:

$$\lambda_i := \frac{w_i}{\sum w_j}, w_i := \frac{\tan\frac{\alpha_{i-1}}{2} + \tan\frac{\alpha_i}{2}}{\text{distance}(v_i, v_0)} \tag{13}$$

See more in *Mean Value Coordinates ,Michael S. Floater*.

Frame Title

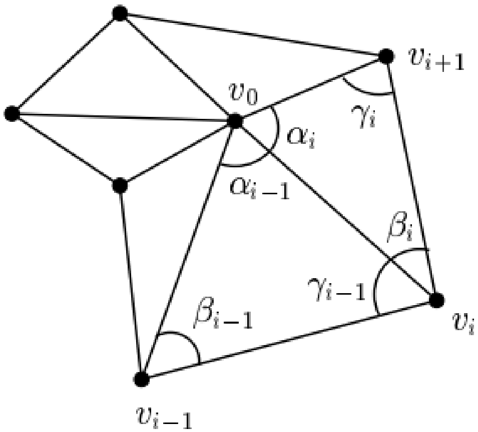


Fig. 7: Star shaped polygon

Representation

- **Parameters in PDE**. Trivial. If the parameter is something like a force density distribution, it can be expressed in a point-wise form or via the coefficients.
- **Geometry of the Boundary**. If $\partial\Omega$ is limited to a simple geometry family, like rectangles, then simple parameters, width and length, are enough to represent. However, when it comes to an irregular one, one need to carefully consider the relation between representations of the geometry and the coordinates.
- **Coordinates**. A proposal is **mean value coordinates** enjoying **equivariance**.
- **Boundary Conditions**. Once the boundary geometry is expressed point-wise, that of the boundary condition is rather simple.

MeshingNet[3]

Target: Given a coarse mesh, the network outputs a refinement regime, **the upper bound of a cell**, more exactly. **A further triangulation** over the coarse grid ensures no cell exceeds the bound. **Loss**: The relative error between the PDE solution over a fine and coarse mesh.

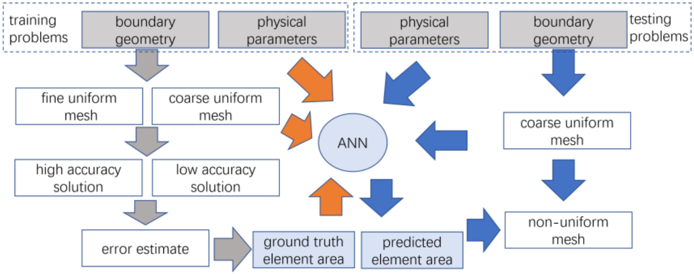


Fig. 8: MeshingNet

Pros and Cons

Advantages

It can generate a refined grid quickly after being pre-trained.

Drawbacks

- It requires lots of data to be trained.
- It takes a good time to train.
- The mesh can't be updated as the system evolves, i.e., **not adaptively**.

M2N[4]:MESH MOVEMENT NETWORKS

It uses **GAT and Spline-NF** to implement a neural operator:

$$F : \mathbf{x}^{t-1}, \text{SystemState} \mapsto \mathbf{x}^{t} \tag{14}$$

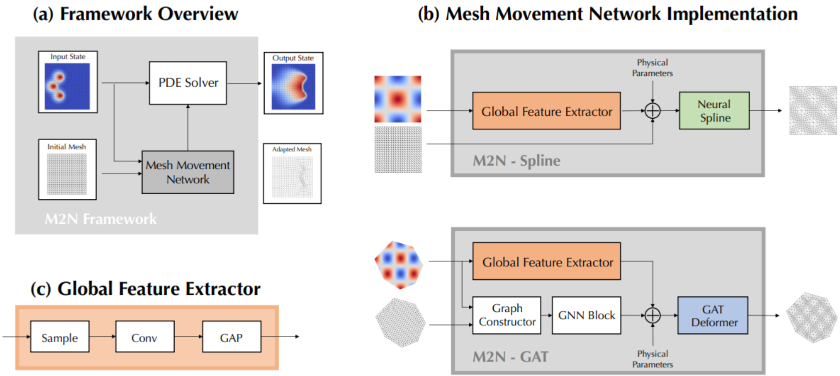


Fig. 9: M2N

Implementation

The spline-NF is simply a flow realized in a rational polynomial field s.t. the transform function on each dimension is monotonic. Now consider that by GAT. A triangulation result can be naturelly viewed as a graph.

- **node feature**. The system state, i.e., the value of the solution u at a point.
- **edge feature**. The **displacement between two nodes**.

To learn more about spline-NF, see *Neural Spline Flow, Conor Durkan et al*.

Section Overview

Sampling

Retrospect PINN

Why is a wiser sampling strategy necessary?

- **Labeled Data Loss**. $MSE_D = \frac{1}{N_d}\sum|\mathbf{u}_\theta(\mathbf{x}_i) - \mathbf{u}_i|^2$.
- **B.C. Loss** $MSE_B = \frac{1}{N_B}\sum|\mathbf{u}_\theta(\mathbf{x}_i, t) - f(\mathbf{x}, t)|^2$.
- **I.C. Loss** $MSE_I = \frac{1}{N_I}\sum|\mathbf{u}_\theta(\mathbf{x}_i, 0) - g(\mathbf{x})|^2$.
- **PDE Loss** $MSE_P = \frac{1}{N_C}\sum|\partial_t\mathbf{u}_\theta + F|^2$.

Importance Sampling[5]

f, q denote original and alternative sampling distributions respectively.

$$\theta^* = \arg\min_{\theta} \mathbb{E}_f [\frac{1}{N} \sum_{j=1}^{N} \mathscr{L}(\theta, x_j)] = \arg\min_{\theta} \mathbb{E}_q [\frac{1}{N} \sum_{j=1}^{N} \frac{f(x_j)}{q(x_j)} \mathscr{L}(\theta, x_j)] \tag{15}$$

No need to really sample again, but modify the gradient descent step :

$$\theta^{(i+1)} = \theta^{(i)} - \eta \sum_{j} \frac{1}{q(x_j)} \nabla_{\theta} \mathscr{L}(\theta, x_j) \tag{16}$$

$q(x_i)$ can be estimated as the ratio of the total loss**[5]**:

$$q(x_j) := \frac{\mathscr{L}(\theta, x_j)}{\sum_j \mathscr{L}(\theta, x_j)} \tag{17}$$

Implementations of Adaptive Importance Sampling[7]

The core idea is the same as **GAN**, which can be written into a **min-max problem**.

$$u^{k+1} = \arg\min_{u} \int \mathscr{L}(\theta, u)^2 q^k(x) dx \tag{18}$$

$$q^{k+1} = \arg \max_{q \geq 0, \int q = 1} -D_{KL}\left(q \| \frac{\mathscr{L}(\theta, u^{k+1})^2}{\int \mathscr{L}(\theta, u^{k+1})^2 dx}\right) \tag{19}$$

[7] uses a GMM instead of a neural network to model q.

Not only Residual(loss)[6]

In practice, we shall also sample those with large gradient and the sensitive boundary. And it is proposed to randomly examine a sub-region when adding new sample locations.

Algorithm 3: ASM III: Residual/gradient/BC-based adaptive sampling method

Step 1 Train the PINNs based on the residual point set $\mathcal{T}$ and the boundary point set $\mathcal{B}$;

Step 2 Divide the computational domain into $\boldsymbol{S}$ subdomains $\Omega_i, i = 1, \ldots, \boldsymbol{S}$, and compute

$$\mathcal{E}_i = \int_{\Omega_i} |R| d\boldsymbol{x}, \quad \mathbf{E}_i = \int_{\Omega_i} |\nabla u| d\boldsymbol{x}$$

using the Gaussian quadrature method for each sub-domain; using the Gaussian quadrature method for each sub-domain;

Step 3 Let $\mathcal{E}_{max} = \max_{i=1,\ldots,\boldsymbol{S}}\{\mathcal{E}_i\}$ and $\mathbf{E}_{max} = \max_{i=1,\ldots,\boldsymbol{S}}\{\mathbf{E}_i\}$. If $\mathcal{E}_{max} < \mathcal{E}_c$, iteration stops. Otherwise, randomly sample $\mathcal{M}$ data in the sub-domain which has the largest $\mathcal{E}$ and $\mathcal{M}$ data in the sub-domain which has the largest $\mathbf{E}$, and then compute the corresponding residuals and gradients for selected points, respectively. Add m_R points which have the largest $|R|$ and $m_{\nabla u}$ points which have the largest $|\nabla u|$ in the residual point set .

Step 4 Randomly sample $\mathcal{M}_{BC}$ boundary points and compute the absolute value of u at these boundary points. Then add m_{BC} points which have the largest $|u|$ in the boundary point set $\mathcal{B}$.

THE END.