Improved Random Feature Method

-PDEs and Machine Learning

Haojun Qin[1]

[1]School of Mathematical Sciences
University of Sciences and Technology of China

May 18th, 2023

Table of Contents

Table of Contents

Project Team Members

- Haojun Qin, USTC
- Xianglong Hou, USTC.
- Bingsong Gao, USTC.
- Lei Zhao, USTC.
- Advisor: Jingrun Chen, Suzhou Institute for Advanced Research, USTC.

Table of Contents

RFM

Consider the following problem:

$$\begin{cases} \mathcal{L}u(x) = f(x), \ x \in \Omega \\ \mathcal{B}u(x) = g(x), \ x \in \partial\Omega \end{cases}$$

The solution is approximated by a linear combination of some trial function:

$$u(x) \approx u_M(x) = \sum_{i=1}^{M} u_i \phi_i(x).$$

$\phi_i(x)$ is called the trial function (generated by neural network).

Trial Function

Define $\phi_i(x)$ as the ones that occur naturally in neural network:

$$\phi_i(x) = \sigma(k_i \cdot x + b_i).$$

where σ is some scalar nonlinear function called activation function, $k_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$ are some random but fixed parameters.

Activation function such as $\sin, \cos, \tanh$ can all be used.

Partition of unity

We start with a set of points $\{x_j\}_{j=1}^{M_p}$, each of which serves as the center for a component in the patition. And $\{d_j\}_{j=0}^{M_p}$ are the diameter of each component in the patiton.

Define ψ^a and ψ^b in 1-dimension case:

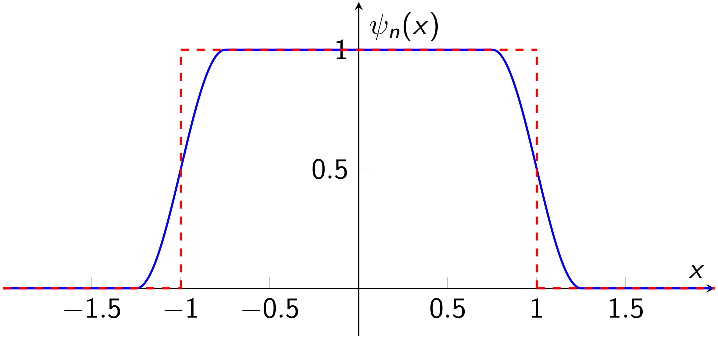

Figure: ψ^a (red) and ψ^b (blue)

Construction of Trial Function

Now we have a partiton of unity consisting M_p components:

$$\{x_j\}_{j=1}^{M_p} + \{d_j\}_{j=1}^{M_p} \Longrightarrow \{\psi_j\}_{j=1}^{M_p}.$$

For each component, we define M original trial functions:

$$\phi_{ij}(x) = \sigma(k_{ij} \cdot x + b_{ij}), \; i = 1, 2, ..., M.$$

Apply ψ_j to ϕ_{ij}:

$$u_M(x) = \sum_{j=1}^{M_p} \sum_{i=1}^{M} u_{ij}\phi_{ij}(x)\psi_j(x)$$

Loss Function

The trial function is:

$$u_M(x) = \sum_{j=1}^{M_p} \sum_{i=1}^{M} u_{ij} \phi_{ij}(x) \psi_j(x)$$

Collect $|C_I|$ points in Ω and $|C_B|$ points on $\partial\Omega$. And define the Loss Function:

$$L = \sum_{x_k \in C_I} \lambda_k \|\mathcal{L} u_M(x_k) - f(x_k)\|_{l^2}^2 + \sum_{x_j \in C_B} \lambda_j \|\mathcal{B} u_M(x_j) - g(x_j)\|_{l^2}^2 .$$

where λ_k are the penalty parameters.

By using least-squares method, we obtain u_{ij} minimizing L.

RFM

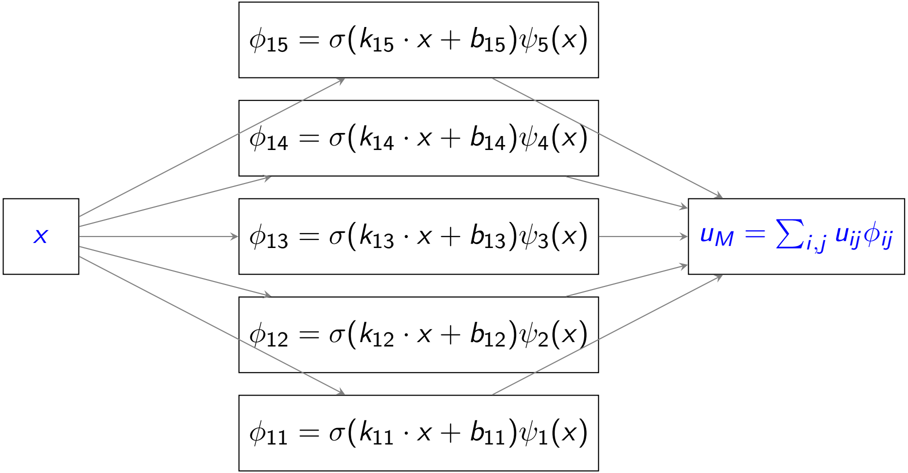

Figure: Random Feature Method

Table of Contents

Reinforced-RFM

Consider a contrained problem and then transform it into quadratic programming:
Original:

$$\min \sum_{x_k \in C_I} \lambda_k \|\mathcal{L} u_M(x_k) - f(x_k)\|_{l^2}^2$$
$$\text{s.t. } \mathcal{B} u_M(x_j) = g(x_j),\ \forall x_j \in C_B$$

QP:

$$\min \|Ax - b\|_{l^2}^2$$
$$\text{s.t. } Cx = d$$

Solve by augmented lagrangian method.

Strict-RFM

Aim: Enforce the exact boundary conditions.
Key Method: Apply distance function to the construction of trial function.

Example

Consider the ODE problem:

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = 0, \ \ y(b) = 0 \end{cases}$$

Define the distance function $L(x) = (x - a)(b - x)$ satisfying $L(a) = 0, L(b) = 0, L(x) \neq 0$ when $x \neq a, b$.

For any trial funtion $\phi(x)$, $\phi(x)L(x)$ always satisfies boundary conditions strictly.

Table of Contents

1-dimension problem

Example (Helmholtz equation)

$$\begin{cases} \frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} - \lambda u(x) = f(x) \quad x \in \Omega, \\ u(0) = c_1, \quad u(8) = c_2. \end{cases}$$

Choose

$u(x) = 4\cos(4(x+\frac{3}{20})) + 5\sin(\sqrt{5}(x+\frac{7}{20})) + 2\sin(\sqrt{3}(x+\frac{1}{20})) + 3\sin(x+\frac{17}{20}) + 2.$

1-dimension problem

M_P	M	Q	RFM		Reinforced-RFM		Strict-RFM	
			L_∞	L^2	L_∞	L^2	L_∞	L^2
4	50	50	1.01E-02	1.18E-02	1.13E-02	1.33E-02	1.63E-02	1.27E-02
8	50	50	2.61E-07	2.92E-07	2.11E-07	2.37E-07	2.99E-07	4.27E-07
16	50	50	1.65E-09	1.74E-09	5.16E-10	4.76E-10	1.18E-09	8.98E-10
32	50	50	3.46E-11	2.55E-11	4.66E-12	5.64E-12	1.72E-11	1.49E-11

Table: Helmholtz equation error

2-dimension problem

Example (Poisson equation)

$$\begin{cases} \Delta u(x,y) = f(x,y) \quad x \in [0,1] \times [0,1], \\ u(x,0) = g_1(x), \quad u(x,1) = g_2(x), \\ u(0,y) = h_1(y), \quad u(1,y) = h_2(x). \end{cases}$$

Choose $u(x,y) = -[1.5\cos(\pi x + 0.4\pi) + 2\cos(2\pi x - 0.2\pi)][1.5\cos(\pi y + 0.4\pi) + 2\cos(2\pi y - 0.2\pi)]$.

2-dimension problem

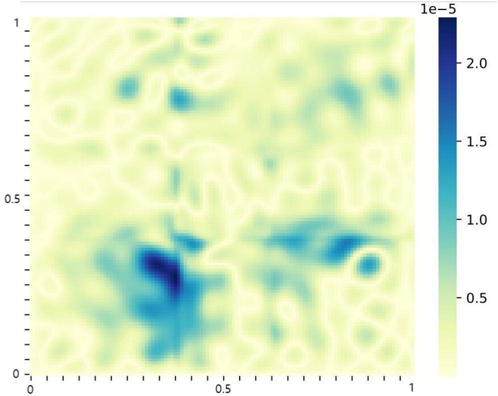

Figure: Poisson equation error

2-dimension problem

M	Q_x	Q_y	RFM		Reinforced-RFM		Strict-RFM	
			L_∞	L^2	L_∞	L^2	L_∞	L^2
200	30	30	6.91E-08	1.60E-08	1.36E-05	4.16E-06	2.24E-09	4.54E-10
200	35	35	1.14E-07	2.57E-08	1.25E-06	2.62E-07	4.56E-09	7.17E-10
200	40	40	1.35E-07	1.76E-08	2.30E-06	6.25E-07	1.20E-08	2.36E-09
300	30	30	9.88E-10	1.45E-10	1.54E-06	4.85E-08	1.90E-10	5.01E-11
300	35	35	6.37E-10	8.31E-11	4.86E-07	1.41E-07	6.10E-11	1.15E-11
300	40	40	2.09E-09	4.00E-10	9.87E-06	2.72E-06	5.80E-11	1.52E-11

Table: Poisson equation error

2-dimension problem

M	Q_x	Q_y	RFM Boundary	Reinforced-RFM Boundary	Strict-RFM Boundary
200	30	30	6.91E-08	3.75E-09	0
200	35	35	7.69E-08	2.53E-10	0
200	40	40	1.23E-07	3.49E-10	0
300	30	30	7.42E-10	1.70E-10	0
300	35	35	6.37E-10	1.20E-10	0
300	40	40	6.56E-10	4.67E-10	0

Table: Poisson boundary error

2-dimension problem

Example (Elasticity equation)

$$
\begin{cases}
\frac{E}{1-\mu^2}\left(u_{xx} + \frac{1-\mu}{2}u_{yy} + \frac{1+\mu}{2}v_{xy}\right) + X = 0 & \text{in } \Omega, \\
\frac{E}{1-\mu^2}\left(v_{yy} + \frac{1-\mu}{2}v_{xx} + \frac{1+\mu}{2}u_{xy}\right) + Y = 0 & \text{in } \Omega, \\
\frac{E}{1-\mu^2}\left(n_x(u_x + \mu v_y) + n_y\frac{1-\mu}{2}(u_y + v_x)\right) = \hat{X} & \text{on } \Gamma_1, \\
\frac{E}{1-\mu^2}\left(n_y(v_y + \mu u_x) + n_x\frac{1-\mu}{2}(v_x + u_y)\right) = \hat{Y} & \text{on } \Gamma_1, \\
u = U & \text{on } \Gamma_2, \\
v = V & \text{on } \Gamma_2.
\end{cases}
$$

Choose

$$
\begin{cases}
u = -\frac{px_2}{6EI}\left[(6L - 3x_1)x_1 + (2+\mu)\left(x_2^2 - \frac{D^2}{4}\right)\right], \\
v = \frac{p}{6EI}\left[3\mu(L - x_1)x_2^2 + (4+5\mu)\frac{D^2x_1}{4} + (3L - x_1)x_1^2\right].
\end{cases}
$$

2-dimension problem

N	Q_x	Q_y	RFM		Reinforced-RFM	
			L_∞	L^2	L_∞	L^2
1	20	20	3.57E-14	7.25E-15	5.69E-10	1.93E-10
1	40	40	7.86E-16	2.35E-16	3.17E-11	1.32E-11
1	60	60	2.47E-16	3.50E-17	5.57E-09	1.53E-09
2	20	20	2.53E-14	1.08E-14	4.28E-10	1.41E-10
2	40	40	3.63E-16	8.87E-17	9.50E-11	3.48E-11
2	60	60	9.27E-17	1.48E-17	6.23E-10	2.57E-10

Table: Elasticity equation u error

2-dimension problem

N	Q_x	Q_y	RFM		Reinforced-RFM	
			L_∞	L^2	L_∞	L^2
1	20	20	2.22E-07	5.68E-08	7.44E-04	1.61E-04
1	40	40	3.18E-08	1.77E-09	3.88E-03	9.79E-04
1	60	60	1.53E-08	1.06E-09	2.45E-05	6.40E-06
2	20	20	2.47E-07	5.32E-08	2.20E-03	5.83E-04
2	40	40	1.90E-08	8.26E-10	5.64E-04	8.47E-05
2	60	60	1.64E-08	3.89E-10	7.42E-05	1.29E-05

Table: Elasticity equation boundary error

Table of Contents

Low Rank

Recall the quadratic programming:

$$\begin{aligned} \min\ & \|Ax - b\|_{l^2}^2 \\ \text{s.t. } & Cx = d \end{aligned}$$

Observation: C is a low-rank matrix.

Low Rank

Row	Column	Rank	Row	Column	Rank
2400	800	150	2400	1200	152
2800	800	149	2800	1200	151
3200	800	149	3200	1200	150

Table: rank(C)

Row	Column	Rank	Row	Column	Rank
160	800	126	640	800	136
320	800	135	800	3200	364
480	800	136	1600	3200	378

Table: rank(C)

Low Rank

QP:

$$
\begin{aligned}
&\min \|Ax - b\|_{l^2}^2 \\
&\text{s.t. } Cx = d
\end{aligned}
$$

- Feasible domain $\{x : \ Cx = d\}$ is an empty set. Without carefully choosing parameters, the augmented lagrangian method could easily get stuck in a loop.
- When $\|Cx - d\|$ decreases, $\|Ax - b\|$ would increase. We have to make a compromise between $\mathcal{L}$ and $\mathcal{B}$.
- When dealing with very large matrix, the unimproved iterative algorithm will increase the error and reduce the efficiency.

Future Plan

Plan:

(1). Prior estimation.

(2). Collocation strategy.

(3). Efficient least-squares algorithm.

Thanks for listening!