Bridging Traditional and Machine Learning-Based Algorithms for Solving Partial Differential Equations: The Random Feature Method

Weinan E

Peking University

Beijing, 2022

Collaborators

- Jingrun Chen, USTC
- Xurong Chi, USTC
- Zhouwang Yang, USTC

Example: Two-dimensional Poisson Equation

1. Strong form: Find $u(x,y) \in C^2(\Omega)$, s.t.

$$
\begin{aligned}
-\Delta u(x,y) &= f(x,y), && \text{in } \Omega \\
u(x,y) &= 0, && \text{on } \partial\Omega
\end{aligned}
$$

or

$$\min_{u(x,y)\in H_0^1(\Omega)} \int_\Omega (\Delta u + f)^2 \,\mathrm{d}x\mathrm{d}y$$

2. Weak form: Find $u(x,y) \in H_0^1(\Omega)$, s.t.

$$\int_\Omega \nabla u \cdot \nabla v \mathrm{d}x\mathrm{d}y = \int_\Omega f v \mathrm{d}x\mathrm{d}y, \quad \forall v \in H_0^1(\Omega)$$

3. Variational form :

$$\min_{u(x,y)\in H_0^1(\Omega)} \int_\Omega \left(\frac{1}{2}|\nabla u|^2 - fu\right) \mathrm{d}x\mathrm{d}y$$

Numerical algorithms

- Low computational cost
- Low human cost
- Robustness and generality

An incomplete list of some of the difficulties we still encounter

- Problems with complex geometry: Stokes flow in porous media
- Kinetic equations: Direct simulation Monte Carlo algorithm
- Multi-scale problems

Outline

Traditional Algorithms

Machine Learning-based Algorithms $M \neq N$

A Bridge Between Traditional and Machine-learning Algorithms

Traditional Algorithms

- Strong form: Finite Difference Method, Spectral Collocation Method, Least Square Method
- Variational form: Ritz Method
- Weak form: Finite Element Method, Spectral (Galerkin) Method, Spectral Element Method, Mesh-free Method, etc

Finite Difference Method

- Discretization of equation → grid points (collocation points)

$$-\Delta u\left(x_{i,j}\right) = f\left(x_{i,j}\right)$$

- Discretization of operator → finite difference

$$\frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} = f_{i,j}$$

- Boundary condition

$$u_{1,j} = u_{m,j} = u_{i,1} = u_{i,n} = 0$$

- Total number of conditions = total number of unknowns

Simple, but not easy to handle complex geometries

Spectral Collocation Method

- Discretization of equation $\rightarrow$ grid points (collocation points)

$$-\Delta u\,(x_{i,j}) = f\,(x_{i,j})$$

- Approximation space: Linear combinations of global polynomials (Lagrange polynomials, Fourier polynomials, Chebyshev polynomials, etc)

$$u_N(x,y) = \sum_{i,j} \phi_{ij}(x,y)u_{ij}$$

- Polynomial basis functions need to satisfy boundary conditions

Spectral accuracy, not easy to handle complex geometries

Ritz Method

- Approximation space: Linear combinations of global basis functions (Polynomials, Trigonometric functions, etc)

$$u_N(x,y) = \sum_{i,j} \phi_{ij}(x,y) u_{ij}$$

- Basis functions need to satisfy boundary conditions
- Variational problem: Numerical integration

$$u_N(x,y) = \underset{v_N(x,y)\in H_0^1(\Omega)}{\arg\min} \int_\Omega \left(\frac{1}{2} |\nabla v_N|^2 - f v_N \right) \mathrm{d}x\mathrm{d}y$$

Not easy to handle complex geometries (boundary conditions and numerical integration)

Finite Element Method

- Mesh generation: Tedious and time-consuming ($\sim$ 70% for solving a PDE problem)
- Basis functions: linear combinations of local piecewise polynomials

$$u_N(x, y) = \sum_{i,j} \phi_{ij}(x, y) u_{ij}$$

- Weak form

$$\int_\Omega \nabla u_N \cdot \nabla v \mathrm{d}x \mathrm{d}y = \int_\Omega f v \mathrm{d}x \mathrm{d}y, \quad \forall v \in V_N$$

- Boundary conditions can be enforced easily

Simple, easy to handle complex geometries, but generating the mesh is not easy

Spectral (Galerkin) Method

- Approximation space: Linear combinations of global polynomials

$$u_N(x,y) = \sum_{i,j} \phi_{ij}(x,y) u_{ij}$$

- Polynomial basis functions need to satisfy boundary conditions
- Weak form

$$\int_\Omega \nabla u_N \cdot \nabla v \mathrm{d}x\mathrm{d}y = \int_\Omega f v \mathrm{d}x\mathrm{d}y, \quad \forall v \in V_N$$

Simple, spectral accuracy, not easy to handle complex regions (boundary conditions, numerical integration)

Spectral Element Method

- Mesh generation
- Approximation space: Linear combination of local higher-degree polynomials (Double summation of order index and element index)

$$u_N(x,y) = \sum_{i,j} \phi_{ij}(x,y) u_{ij}$$

- Boundary conditions can be implemented easily
- Weak form:

$$\int_\Omega \nabla u_N \cdot \nabla v \mathrm{d}x\mathrm{d}y = \int_\Omega f v \mathrm{d}x\mathrm{d}y, \quad \forall v \in V_N$$

Spectral accuracy, easy to handle complex geometries
Mesh generation, boundary conditions and numerical integration can be difficult

Meshfree Method

- Approximation space: Linear combinations of global and local functions $u_N(x, y) = \sum_{i,j} \phi_{ij}(x, y) u_{ij}$
- Boundary conditions are enforced by a penalty term
- Weak form:

$$\int_\Omega \nabla u_N \cdot \nabla v \mathrm{d}x\mathrm{d}y = \int_\Omega f v \mathrm{d}x\mathrm{d}y, \quad \forall v \in V_N$$

Simple, algebraic accuracy, not easy to handle complex geometries (numerical integration)

Accuracy vs Efficiency

WHICH METHOD IS BETTER???

- Strong form: Collocation points
- Weak form: Numerical integration
- Approximation space
- Boundary conditions

Note that we always have $M = N$, where

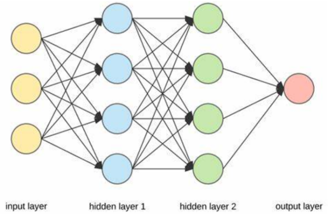
- M = number of parameters
- N = number of equations, or collocation points

Deep Neural Network

A new approximation space

$$u(x,y) = W\sigma\left(W_2\sigma\left(W_1\mathbf{x} + b_1\right) + b_2\right)$$

How to optimize the parameters W and b?



- Strong form: Collocation points
- Variational form: Numerical integration or Monte-Carlo sampling
- Weak form: Numerical integration or Monte-Carlo sampling

Components of a machine-learning algorithm

- Loss function: Strong, variational, weak
  Collocation point, Quadrature or Monte Carlo sampling
- Approximation space: Deep neural networks
- Optimization of NN parameters: Stochastic gradient descent method

Comparison

- Error sources: approximation, integration, optimization
- SGD can get a reasonable solution, which is not good enough
- In high dimension
  - Traditional methods fail
  - Deep learning methods work (1% relative error without convergence order)
- In low dimension $d \leq 3$
  - Traditional methods typically work well
  - Deep-learning methods work (1% relative error without convergence order), but have high coding efficiency

Machine Learning-based Algorithms

- Variational form: Deep Ritz Method (DRM)[1]
- Strong form: Deep Galerkin Method (DGM)[2], Physics-Informed Neural Networks (PINN)[3]
- Weak form: Weak Adversarial Network (WAN)[4]
- etc

[1] **EY2018**.
[2] **SS2018**.
[3] **PINN**.
[4] **Bao**.

Deep Ritz Method

$$\begin{cases} -\Delta u(x) = f(x), & x \in \Omega \\ u(x) = g(x), & x \in \partial\Omega \end{cases}$$

Loss function: Variational form + boundary penalty term

$$I[u] = \int_\Omega \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) \mathrm{d}x + \lambda \int_{\partial\Omega} (u(x) - g(x))^2 \mathrm{d}x$$

Optimization:

$$\begin{aligned} \theta_{k+1} = & \theta_k - \alpha \nabla_\theta \frac{|\Omega|}{N_v} \sum_{i=1}^{N_v} \left[\frac{1}{2} \left| \nabla u\left(x_i\right) \right|^2 - f\left(x_i\right) u\left(x_i\right) \right] \\ & - \lambda \alpha \nabla_\theta \frac{|\partial\Omega|}{N_b} \sum_{j=1}^{N_b} \left[u\left(y_j\right) - g\left(y_j\right) \right]^2 \end{aligned}$$

- Variational form: ReLU converges in general
- Boundary condition is enforced by penalty term, but the penalty parameter is difficult to tune
- Loss function can be negative
- $M \neq N$

DGM, PINN

$$
\begin{aligned}
&\partial_t u = \mathcal{L}u, \quad (t,x) \in [0,T] \times \Omega \\
&u(0,x) = u_0(x), x \in \Omega \\
&u(t,x) = g(x), \quad (t,x) \in [0,T] \times \partial\Omega
\end{aligned}
$$

Loss function: strong form in the least-squares sense + boundary penalty term

$$
\begin{aligned}
L(u) = &\|\partial_t u - \mathcal{L}u\|_{2,[0,T]\times\Omega}^2 + \lambda_1 \|u(0,\cdot) - u_0\|_{2,\Omega}^2 \\
&+ \lambda_2 \|u - g\|_{2,[0,T]\times\partial\Omega}^2
\end{aligned}
$$

- Strong form: High regularity, and usually ReLU does not converge
- Boundary condition is enforced by penalty term, but the penalty parameter is difficult to tune
- $M \neq N$

WAN

$$\begin{cases} \langle \mathcal{A}[u], \varphi \rangle \triangleq \int_\Omega \left(\sum_{j=1}^d \sum_{i=1}^d a_{ij} \partial_j u \partial_i \varphi + \sum_{i=1}^d b_i \varphi \partial_i u + cu\varphi - f\varphi \right) \mathrm{d}x = 0 \\ \mathcal{B}[u] = 0, \quad \text{on } \partial\Omega \end{cases}$$

Loss function: weak form

$$\|\mathcal{A}[u]\|_{op} \triangleq \max \left\{ \langle \mathcal{A}[u], \varphi \rangle / \|\varphi\|_2 \mid \varphi \in H_0^1, \varphi \neq 0 \right\}$$

$$\min_{u \in H^1} \|\mathcal{A}[u]\|_{\mathrm{op}}^2 \iff \min_{u \in H^1} \max_{\varphi \in H_0^1} |\langle \mathcal{A}[u], \varphi \rangle|^2 / \|\varphi\|_2^2$$

$$L_{\text{int}}(\theta, \eta) \triangleq \log \left| \langle \mathcal{A}[u_\theta], \varphi_\eta \rangle \right|^2 - \log \|\varphi_\eta\|_2^2$$

$$L_{\text{bdry}}(\theta) \triangleq (1/N_b) \cdot \sum_{j=1}^{N_b} \left| u_\theta\left(x_b^{(j)}\right) - g\left(x_b^{(j)}\right) \right|^2$$

$$\min_\theta \max_\eta L(\theta, \eta), \qquad \text{where } L(\theta, \eta) \triangleq L_{\text{int}}(\theta, \eta) + \alpha L_{\text{bdry}}(\theta)$$

- Weak form: ReLU converges in general
- Boundary conditions require penalty terms
- Min-max problem: uses GAN to solve and takes longer to optimize
- $M \neq N$

Machine Learning-based Algorithms

- Simple, meshfree, easy to handle complex geometries and boundary conditions
- The accuracy cannot be improved systematically and the penalty parameters are difficult to tune
- Training takes a long time and the optimization error is difficult to quantify
- Low human cost and low application barrier

Local Extreme Learning Machine[6]

- Strong form: collocation points
- Approximation space: domain decomposition + extreme learning machine (only parameters in the output layer optimized)[5]
- Linear least-squares problem $M \neq N$
- Similar to the spectral element method

Spectral accuracy, easy to handle complex geometries

[5] **huang2006extreme**.
[6] **dong2021local**.

- Scalar PDE form

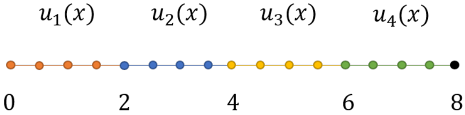
$$\begin{cases} \mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}), \text{ in } \Omega \\ \mathcal{B}u(\boldsymbol{x}) = g(\boldsymbol{x}), \text{ on } \partial\Omega \end{cases}$$

- Domain decomposition: $\Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_{N_e}$
- Local neural network is used to represent the solution in each subdomain
- Continuity conditions of basis functions and derivatives are enforced
- Main steps in the algorithm:
  1. Selecting collocation points in subdomains Ω_s
  2. Evaluating the equations at interior points and boundary/continuity conditions at (sub-)boundary points
  3. Solving the least-squares problem

Illustration

Domain $[0, 8]$ with $N = 4$ subdomains

- Equation at all points
- Boundary conditions at $x = 0$ and $x = 8$
- Continuity conditions at $x = 2$, $x = 4$ and $x = 6$



Exponential convergence for Helmholtz equation

N	L^∞ error	L^2 error
4	8.76E-2	2.31E-2
8	4.06E-7	1.20E-7
16	3.52E-10	1.14E-10
32	1.73E-11	5.99E-12

Timoshenko beam: Loss of exponential accuracy

$N_x * N_y$	$Q_x * Q_y$	u error	v error	σ_x error	τ_{xy} error
2*2	5*5	5.22E-3	4.90E-3	1.33E-2	2.39E-2
	10*10	1.55E-4	5.25E-5	1.44E-4	1.02E-4
	20*20	6.36E-4	3.47E-4	6.55E-4	7.26E-4
	40*40	1.76E-3	1.64E-3	1.93E-3	2.57E-3
4*4	5*5	8.50E-2	4.04E-2	7.72E-2	4.19E-2
	10*10	1.32E-5	6.19E-6	3.25E-5	4.22E-5
	20*20	1.33E-3	1.12E-3	1.31E-3	1.04E-3
	40*40	6.42E-4	1.91E-4	1.18E-3	1.38E-3

Comparison with Spectral Element Method

Local ELM	SEM
Strong form	Weak form
Domain decomposition	Mesh generation
Extreme learning machine	Polynomial
$M \neq N$	$M = N$
Spectral accuracy	Spectral accuracy
Geometry more friendly	Geometry friendly
Basis do not satisfy BC	Basis satisfy BC

Local ELM does not work well for anisotropy/elasticity problems

Accuracy vs Efficiency

Is there a way to combine the advantages of traditional and machine learning-based methods?

The Random Feature Method (RFM)[7]

- Strong form: collocation points
- Approximation space: random feature functions
  1. Partition of unity and local random feature models
  2. Multi-scale basis
  3. Adaptive basis
- Soft boundary condition: Basis functions do not satisfy BC
- A linear convex optimization problem with easy-tuning parameters (balance the contributions from the PDE terms and the boundary conditions in the loss function)
- $M \neq N$

Simple, mesh-free, spectral accuracy, easy to handle complex geometries and boundary conditions

[7] **RFM**.

Loss function

Examples include the elliptic problem, the linear elasticity problem, and the Stokes flow problem when $d \le 3$

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

where $\boldsymbol{x} = (x_1, \cdots, x_d)^T$, and Ω is bounded and connected domain in $\mathbb{R}^d$

$$Loss = \sum_{\boldsymbol{x}_i \in C_I} \sum_{k=1}^{K_I} \lambda_{Ii}^k \|\mathcal{L}^k \boldsymbol{u}(\boldsymbol{x}_i) - \boldsymbol{f}^k(\boldsymbol{x}_i)\|_{l^2}^2 + \sum_{\boldsymbol{x}_j \in C_B} \sum_{\ell=1}^{K_B} \lambda_{Bj}^\ell \|\mathcal{B}^\ell \boldsymbol{u}(\boldsymbol{x}_j) - \boldsymbol{g}^\ell(\boldsymbol{x}_j)\|_{l^2}^2$$

Different penalty parameters at different collocation points are allowed

Collocation points

Two sets of collocation points: C_I in Ω and C_B on $\partial\Omega$

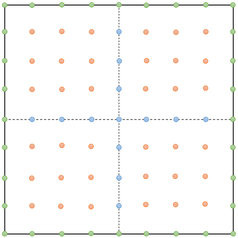


Figure: Collocation points for a square domain: C_I, interior points in orange and blue; C_B, boundary points in green.

Approximation space

A linear combination of M network basis functions $\{\phi_m\}$ over Ω as

$$u_M(\boldsymbol{x}) = \sum_{m=1}^{M} u_m \phi_m(\boldsymbol{x})$$

$$\phi_m(\boldsymbol{x}) = \sigma(\boldsymbol{k}_m \cdot \boldsymbol{x} + b_m)$$

where σ is some scalar nonlinear function, $\boldsymbol{k}_m, b_m$ are some random but fixed parameters

Partition of unity

A set of points $\{\boldsymbol{x}_n\}_{n=1}^{M_p} \subset \Omega$ with $\boldsymbol{x}_n$ the center for a component in the partition

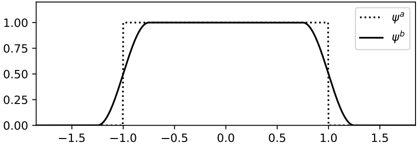


Figure: Visualization of $\psi^a(x)$ and $\psi^b(x)$.

High-dimensional PoU: $\psi_n(\boldsymbol{x}) = \prod_{k=1}^{d} \psi_n(x_k)$

Local random feature functions

$$\tilde{\boldsymbol{x}} = \frac{1}{\boldsymbol{r}_n}(\boldsymbol{x} - \boldsymbol{x}_n), \quad n = 1, \cdots, M_p$$

where $\boldsymbol{r}_n = (r_{n1}, r_{n2}, \cdots, r_{nd})$ and $\{\boldsymbol{r}_n\}$ are preselected

- Construct J_n random feature functions by

$$\phi_{nj}(\boldsymbol{x}) = \sigma(\boldsymbol{k}_{nj} \cdot \tilde{\boldsymbol{x}} + b_{nj}), \quad j = 1, \cdots, J_n$$

  where the feature vectors $\{(\boldsymbol{k}_{nj}, b_{nj})\}$ are often chosen randomly, such as $\boldsymbol{k}_{nj} \sim \mathbb{U}([-R_{nj}, R_{nj}]^d)$ and $b_{nj} \sim \mathbb{U}([-R_{nj}, R_{nj}])$

- Approximate solution

$$u_M(\boldsymbol{x}) = \sum_{n=1}^{M_p} \psi_n(\boldsymbol{x}) \sum_{j=1}^{J_n} u_{nj}\phi_{nj}(\boldsymbol{x})$$

Multi-scale basis

$$u_M(\boldsymbol{x}) = u_g(\boldsymbol{x}) + \sum_{n=1}^{M_p} \psi_n(\boldsymbol{x}) \sum_{j=1}^{J_n} u_{nj}\phi_{nj}(\boldsymbol{x})$$

where u_g is a global random feature function

Adaptive basis

- Some (incomplete) information about the spectral distribution of the solution in the precomputing stage
- A spectral analysis of the forcing term for example
- Selection of the spectral distribution of the feature vectors
- Particularly useful when $\sin/\cos$ is used as the activation function

Optimization: A least-squares problem

Parameter tuning is fully automatic!!!

Penalty coefficients in the loss functions are chosen as

$$\lambda_{Ii}^{k} = \frac{c}{\max\limits_{1\le n\le M_p}\max\limits_{1\le j'\le J_n}\max\limits_{1\le k'\le K_I}|\mathcal{L}^k(\phi_{nj'}^{k'}(\boldsymbol{x}_i)\psi_n(\boldsymbol{x}_i))|} \quad \boldsymbol{x}_i \in C_I,\ k = 1,\cdots,K_I$$

$$\lambda_{Bj}^{\ell} = \frac{c}{\max\limits_{1\le n\le M_p}\max\limits_{1\le j'\le J_n}\max\limits_{1\le \ell'\le K_I}|\mathcal{B}^\ell(\phi_{nj'}^{\ell'}(\boldsymbol{x}_j)\psi_n(\boldsymbol{x}_j))|} \quad \boldsymbol{x}_j \in C_B,\ \ell = 1,\cdots,K_B$$

where $c = 100$ is a universal constant

Collocation points

- Explicit representation of boundary
  - Uniform grid over the computational domain
  - Uniform grid in the parameter space
- Implicit representation of boundary
  - Easily identify interior points
  - Define an energy function for finding a point on the boundary

Numerical setup

- Select a set of points $\{\boldsymbol{x}_n\}_{n=1}^{M_p}$ and construct the PoU
- Construct J_n random feature functions with radius $\boldsymbol{r}_n$ for each $\boldsymbol{x}_n$
- Sample Q collocation points
- Total number of random feature functions M
- Total number of conditions N
- Typically $N > M$ due to the geometric complexity and the limited computational resource

Partition of unity and local random feature models

Table: Comparison of the RFM and PINN for the one-dimensional Helmholtz equation

M	ψ^a		ψ^b		PINN	
	N	L^∞ error	N	L^∞ error	N	L^∞ error
200	208	8.76E-2	202	2.51E-2	202	2.59E-2
400	416	5.89E-7	402	5.18E-7	402	6.77E-3
800	832	4.44E-10	802	6.61E-10	802	1.35E-2
1600	1664	8.84E-12	1602	1.18E-11	1602	8.94E-3

- Error in PINN is around $1E-3$ without notable further improvement $\leftarrow$ Optimization error
- RFM for different PoU functions has exponential convergence $\leftarrow$ representability of random feature functions
- RFM has exponential convergence for all problems tested when $d = 1, 2, 3$

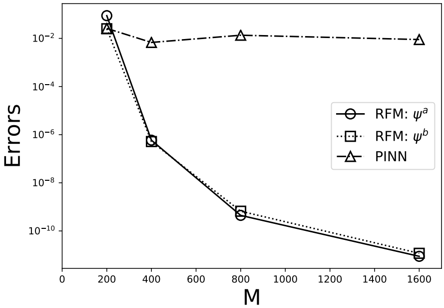
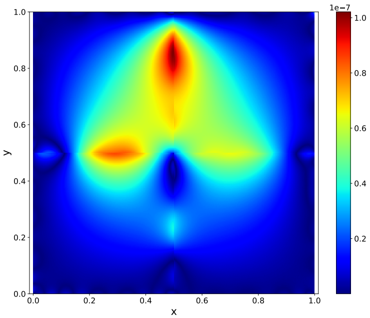
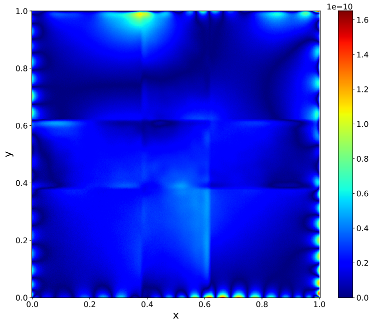


Figure: Convergence of RFM and PINN for Helmholtz equation in the semi-log scale

Different choice of PoU



(a) ψ^a



(b) ψ^b

Figure: Error distribution of the RFM with different choices of PoU for Poisson equation

Multi-scale basis

Table: Comparison of PoU-based local basis and multi-scale basis functions for Poisson equation with the explicit solution

Solution frequency	M	N	PoU-based basis	Multi-scale basis
Low	1200	1920	1.93E-8	3.28E-9
	2700	4320	3.62E-9	6.42E-10
	4800	7680	8.61E-10	3.05E-10
High	1200	1920	6.42E-6	9.36E-7
	2700	4320	1.34E-7	3.58E-8
	4800	7680	4.16E-8	1.75E-8
Mixed	1200	1920	3.22E-6	4.68E-7
	2700	4320	6.54E-8	1.80E-8
	4800	7680	2.06E-8	8.92E-9

Inclusion of global basis functions improves the accuracy when the solution has a significant low-frequency component

Adaptive basis

Table: Results of using adaptive random feature functions for the two-dimensional Poisson equation

R_m	tanh		sin	
	$\mathbb{U}[-R_m, R_m]$	Equally spaced	$\mathbb{U}[-R_m, R_m]$	Equally spaced
0.5	4.92E-9	1.01E-9	2.55E-3	6.05E-4
1.0	2.91E-8	9.36E-9	8.96E-7	2.58E-5
1.5	1.33E-6	5.95E-7	1.79E-9	1.47E-6
2.0	8.75E-5	7.85E-5	3.30E-12	4.29E-7
2.5	8.16E-4	4.70E-5	2.86E-12	7.66E-6
3.0	2.06E-2	5.27E-4	7.32E-12	2.17E-5
3.5	1.53E-3	3.95E-3	6.10E-12	7.45E-5
4.0	2.66E-3	1.27E-3	6.10E-12	5.59E-5
4.5	5.39E-3	1.76E-2	2.29E-11	1.24E-3
5.0	1.29E-2	5.16E-2	2.17E-11	6.72E-3

Best results: sin activation function with $R_m \geq k$ and random initialization

Timoshenko beam problem: Elasticity problem in two dimension

Table: Comparison of RFM and locELM

Method	M	N	u error	v error	σ_x error	τ_{xy} error
RFM	800	400	1.36E-2	3.43E-3	1.40E-2	1.63E-2
		1200	7.14E-6	7.98E-7	8.93E-6	7.45E-6
		4000	6.41E-11	4.34E-11	6.41E-11	6.58E-11
		14400	8.16E-12	1.01E-12	1.07E-11	1.03E-11
locELM	800	400	5.22E-3	4.90E-3	1.33E-2	2.39E-2
		1200	1.55E-4	5.25E-5	1.44E-4	1.02E-4
		4000	6.36E-4	3.47E-4	6.55E-4	7.26E-4
		14400	1.76E-3	1.64E-3	1.93E-3	2.57E-3

Rescaling strategy restores the spectral accuracy

Two-dimensional elasticity problem with a complex geometry

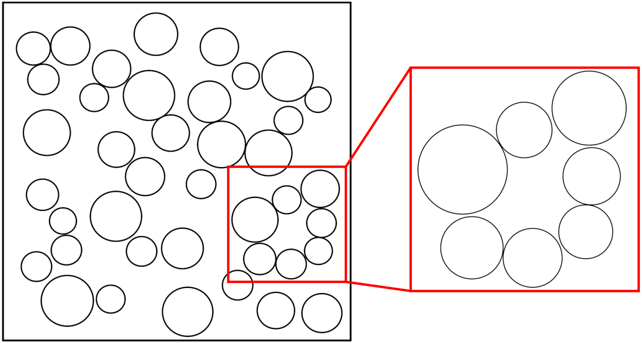
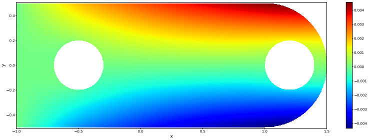


Figure: Complex domain with a cluster of holes that are nearly touching

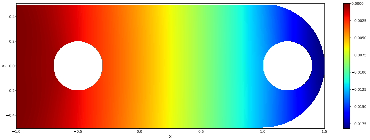
Rescaling

Error in locELM is around $10^{-3} \sim 10^{-2}$, while RFM still maintains spectral accuracy

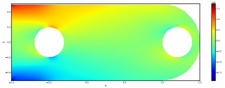
M	N	u error	v error	σ_x error	σ_y error	τ_{xy} error
3200	1784	4.96E-1	8.37E-1	1.09E+0	3.52E+0	5.24E-1
	4658	5.82E-3	7.12E-3	1.04E-2	5.47E-2	3.85E-3
	13338	1.69E-5	1.19E-5	2.89E-5	6.40E-5	8.18E-6
	42820	1.39E-5	1.55E-5	4.92E-5	6.16E-5	1.29E-5
12800	6578	9.11E-2	6.41E-2	1.03E-1	2.46E-1	2.95E-2
	17178	2.35E-4	2.10E-4	3.02E-4	7.56E-4	8.93E-5
	50500	5.46E-7	4.98E-7	8.45E-7	2.03E-6	2.67E-7
	165184	2.32E-7	1.89E-7	9.28E-8	2.32E-7	2.43E-8



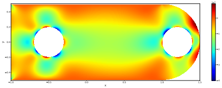
(a) u



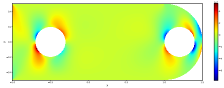
(b) v



(c) σ_x



(d) τ_{xy}



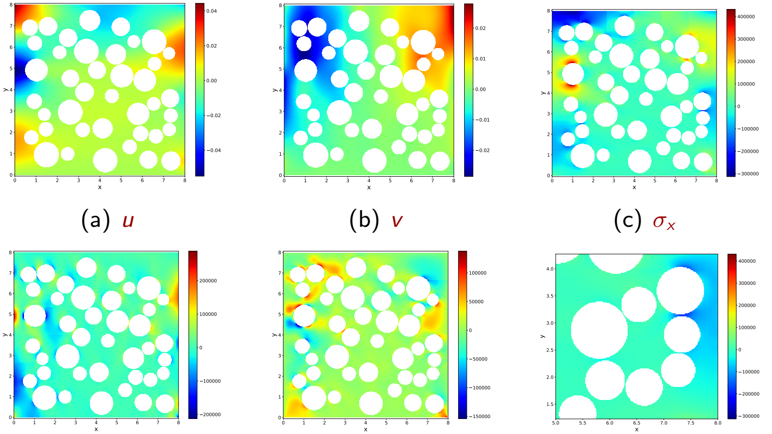
(e) σ_y

Figure: Numerical solution by the random feature method for the elasticity problem

Difference between the RFM and FEM solutions is about 1%

Method	Reference	M	N	u error	v error	σ_x error	σ_y error	τ_{xy} error
RFM	RFM $N = 490176$	16000	40326	1.28E+0	1.12E+0	1.29E+0	9.37E-1	1.03E+0
			135442	1.12E-1	1.16E-1	1.13E-1	1.03E-2	1.20E-1
			285472	6.52E-4	6.98E-4	1.03E-3	3.01E-5	1.88E-3
RFM	FEM $M = 153562$	16000	40326	1.30E+0	1.12E+0	1.28E+0	9.37E-1	1.03E+0
			135442	7.65E-2	8.55E-2	1.16E-1	1.31E-1	1.25E-1
			285472	3.94E-2	3.36E-2	6.59E-3	5.95E-2	2.31E-2
			490176	4.00E-2	3.43E-2	6.20E-3	5.92E-2	2.30E-2
FEM	FEM $M = 153562$	3716	3716	3.15E-4	4.54E-4	1.41E-2	5.81E-2	3.35E-2
		10438	10438	1.20E-4	1.81E-4	9.39E-3	3.61E-2	2.13E-2
		40054	40054	2.88E-5	3.93E-5	4.65E-3	1.62E-2	9.40E-3
FEM	RFM $N = 490176$	3716	3716	3.87E-2	3.36E-2	1.43E-2	8.93E-2	3.86E-2
		10438	10438	3.86E-2	3.34E-2	1.05E-2	7.29E-2	2.99E-2
		40054	40054	3.85E-2	3.32E-2	7.19E-3	6.33E-2	2.44E-2
		153562	153562	3.85E-2	3.32E-2	6.22E-3	6.01E-2	2.31E-2

Table: Comparison of RFM and FEM

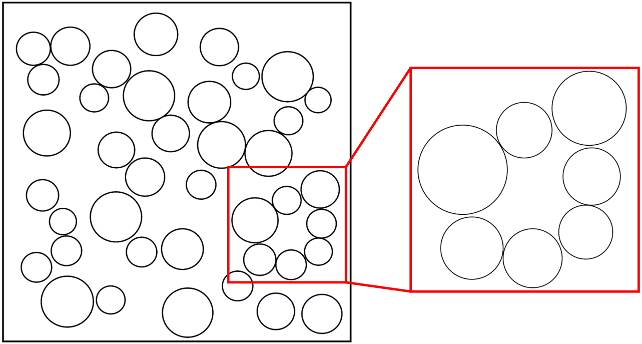


(a) u (b) v (c) σ_x

(d) σ_y (e) τ_{xy} (f) σ_x over a cluster

Figure: Numerical solution by the random feature method for the two-dimensional elasticity problem over a complex geometry

Mesh generation in FEM is difficult

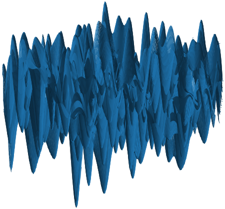


Removal of the cluster leads to an L^∞ error of about 50% for σ_x
RFM shows a clear trend of numerical convergence

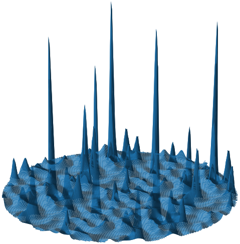
M	N	u error	v error	σ_x error	σ_y error	τ_{xy} error
14400	195146	2.30E-1	1.30E-1	6.64E-2	1.72E-1	1.71E-1
	226132	8.97E-2	1.23E-1	5.60E-2	1.41E-1	1.32E-1
	259400	6.47E-2	6.94E-2	3.66E-2	9.04E-2	8.15E-2
	294878	7.30E-2	6.68E-2	3.46E-2	7.13E-2	7.05E-2

Table: Numerical results of the RFM with $N = 332606$ as the reference

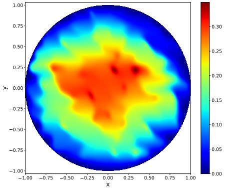
Multi-scale problems



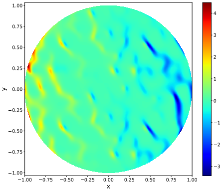
(a) h



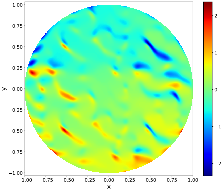
(b) a



(c) u



(d) u_x



(e) u_y

Figure: Random feature method for the elliptic homogenization problem

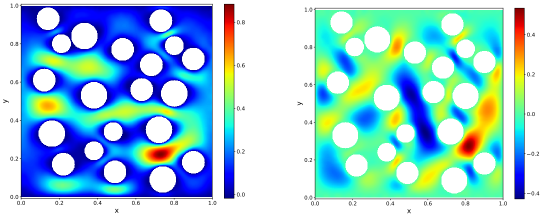
Table: Convergence of RFM

M	N	u error	u_x error	u_y error
25600	25554	1.42E+0	8.68E+0	8.73E+0
	91339	3.13E-2	3.54E-2	3.62E-2
	197360	3.48E-3	6.45E-3	7.18E-3
	343586	Reference		

Stokes flow

Two-dimensional channel flows with the inhomogeneous boundary condition

$$(u,v)|_{\partial\Omega} = \begin{cases} (y(1-y),0) & \text{if } x=0 \\ (y(1-y),0) & \text{if } x=1 \\ (0,0) & \text{otherwise} \end{cases}$$

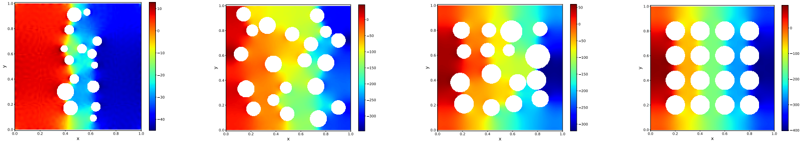


(a) u (b) v

Figure: Velocity field (u,v) generated by the random feature method

Pressure diagram for four sets of complex obstacles

- Spurious pressure mode arises due to the rank deficiency of the discrete systems in spectral methods $M = N$[8]
- RFM automatically bypass this issue by looking for the minimal-norm solution $M \neq N$



(a) (b) (c) (d)

[8] **schumack1991spectral**.

Discussions

Three key components of RFM

1. Loss function: least-squares (strong) formulation of the PDEs on collocation points
2. Approximate solution: linear combination of random feature functions
3. Optimization: least-squares problem with automatic parameter tuning

- Traditional algorithms are robust but lack of flexibility
- Machine-learning algorithms are flexible but lack of robustness
- RFM seems to have both

- Deep neural networks have strong representative power but the parameters are difficult to optimize
- Random feature functions seem to also have strong representative power and the parameters are "easy" to optimize

- Classical methods $M = N$: Efficient linear solvers
- Random feature method $M \neq N$: Least square framework with large condition number

Further developments

- Choice of basis functions: Probability distribution for the feature vector
- Choice of collocation points: Three dimensional domains when the boundary is a surface
- Training: Preconditioning and reformulation techniques
- Time-dependent problems
- Applications