Deep Learning and Numerical PDEs

Iterative Methods and Frequency Principle

Jinchao Xu

KAUST and Penn State

xu@multigrid.org

Morgan State University, June 20th, 2023

CBMS Lecture Series

NSF DMS-2111387, KAUST Baseline Research Fund

Table of Contents

A fundamental problem in scientific computing

Given $A \in R^{N\times N}$, $b \in R^N$, how to solve $Ax = b$ efficiently?

Issue: cost! (it often takes 60–99% of the whole simulation time!)

Oldest method: Gaussian elimination

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases}$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ \qquad\quad \tilde{a}_{22}x_2 + \tilde{a}_{23}x_3 = \tilde{b}_2 \\ \qquad\quad \tilde{a}_{32}x_2 + \tilde{a}_{33}x_3 = \tilde{b}_3 \end{cases}$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ \qquad\quad \tilde{a}_{22}x_2 + \tilde{a}_{23}x_3 = \tilde{b}_2 \\ \qquad\qquad\qquad \bar{a}_{33}x_3 = \bar{b}_3 \end{cases}$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \end{cases}$$

Variational Principle

Variational principle:

$$w(x) \equiv 0 \quad \Longleftrightarrow \int_0^1 w(x)v(x)\,dx = 0 \quad \forall v$$

Variational formulation for 1D elasticity equation

1D linear elasticity equation on $[0,1]$

$$-u'' = f \quad u(0) = u(1) = 0. \tag{1}$$

Consider:

$$V = \{v : \text{continuous and piecewise smooth on } [0,1],\ v(0) = v(1) = 0\} \tag{2}$$

and integrate by parts

$$\int_0^1 -u''v\,dx = \int_0^1 u'v'\,dx + u'v\Big|_0^1 = \int_0^1 u'v'\,dx.$$

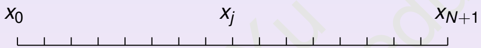
Variational formulation: Find $u \in V$

$$\int_0^1 u'v'\,dx = \int_0^1 fv\,dx \quad \forall v \in V. \tag{3}$$

1D Finite element space

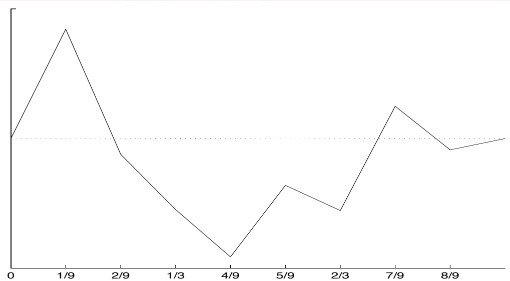
- Uniform grid $\mathcal{T}_h$

$$0 = x_0 < x_1 < \cdots < x_{N+1} = 1, \quad x_j = \frac{j}{N+1} \ (j = 0 : N+1).$$



- Linear finite element space

$$V_h = \{v : \ v \text{ is continuous and piecewise linear w.r.t. } \mathcal{T}_h, \ v(0) = v(1) = 0\}.$$



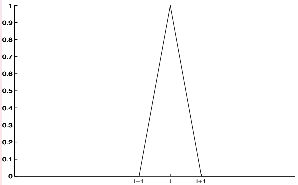
Galerkin method

- Galerkin method: Find $u_h \in V_h$ such that

$$\int_0^1 u_h' v_h' \, dx = \int_0^1 f v_h \, dx \quad \forall v_h \in V_h.$$

- $u_h = \sum_{i=1}^N u_i \varphi_i(x)$
- Nodal basis: $\varphi_i(x_j) = \delta_{ij}$

$$\varphi_i(x) = \begin{cases} \frac{x - x_{i-1}}{h}, & x \in [x_{i-1}, x_i]; \\ \frac{x_{i+1} - x}{h}, & x \in [x_i, x_{i+1}]; \\ 0 & \text{elsewhere.} \end{cases}$$



1D linear system on uniform grid

- Stiffness matrix $a_{ij} = \int_0^1 \varphi_j' \varphi_i' \, dx$, $b_i = \int_0^1 f \varphi_i \, dx$

$$A = \frac{1}{h} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \quad b = h \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{pmatrix} + \mathcal{O}(h^3).$$

Iterative methods for $Au = f$

$$u^0, u^1, \dots, u^{m-1} \longrightarrow u^m$$

Basic ideas:

1. Form the residual: $r = f - Au^{m-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^m = u^{m-1} + \hat{e}$

Linear iterative method:

$$u^m = u^{m-1} + B(f - Au^{m-1}) \tag{4}$$

Let $A = L + D + U$. Thus,

- Jacobi iteration: $B = D^{-1}$,
- Gauss-Seidel iteration: $B = (L + D)^{-1}$.

Examples: basic iterative methods

- Richardson iteration:

$$u^m = u^{m-1} + \omega(f - Au^{m-1}), \quad m = 1, 2, \cdots, \tag{5}$$

- Modified Jacobi:

$$u^m = u^{m-1} + \omega D^{-1}(f - Au^{m-1}), \quad m = 1, 2, \cdots, \tag{6}$$

- Modified Gauss-Seidel:

$$u^m = u^{m-1} + (\omega^{-1}D + L)^{-1}(f - Au^{m-1}), \quad m = 1, 2, \cdots, \tag{7}$$

Thus, the iterative method converges if the following operator is SPD:

$$(B')^{-1} + B^{-1} - A = \begin{cases} 2\omega^{-1} - A > 0 & \text{if } 0 < \omega < \dfrac{2}{\rho(A)} & \text{Richardson;} \\ 2\omega^{-1}D - A > 0 & \text{if } 0 < \omega < \dfrac{2}{\rho(D^{-1}A)} & \text{Modified Jacobi;} \\ (2-\omega)\omega^{-1}D > 0 & \text{if } 0 < \omega < 2 & \text{Modified G.-S.} \end{cases}$$

Iterative methods: Gauss–Seidel

Consider a simple algebraic system:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned}$$

Gauss-Seidel method ($x^{m-1} \to x^m$):

$$\begin{aligned} a_{11}x_1^m + a_{12}x_2^{m-1} + a_{13}x_3^{m-1} &= b_1 \\ a_{21}x_1^m + a_{22}x_2^m + a_{23}x_3^{m-1} &= b_2 \\ a_{31}x_1^m + a_{32}x_2^m + a_{33}x_3^m &= b_3 \end{aligned}$$

- It converges for any symmetric, positive and definite (SPD) system.
- Only involves the inversion of the diagonal elements: a_{ii}^{-1}
- However, if converges very slowly if the linear system is large.

Iterative methods as gradient descent (GD)

If A is SPD, then we have the following equivalence:

$$Au = f \quad \Longleftrightarrow \quad \min \underbrace{\frac{1}{2}u^T A u - f^T u}_{J(u)}$$

- Richardson for $Au = f \Leftrightarrow$ Gradient descent for $f(u)$

$$u^m = u^{m-1} + \eta(f - Au^{m-1}) = u^{m-1} - \eta \nabla J(u^{m-1})$$

- Jacobi for $Au = f \Leftrightarrow$ Scaled gradient descent for $f(u)$

$$u^m = u^{m-1} + \eta D^{-1}(f - Au^{m-1}) = u^{m-1} - \eta [\mathrm{diag}(A)]^{-1} \nabla J(u^{m-1})$$

- Gauss–Seidel for $Au = f \Leftrightarrow$ Preconditioned gradient descent for $f(u)$

$$u^m = u^{m-1} + (\eta D + L)^{-1}(f - Au^{m-1}) = u^{m-1} - P\nabla J(u^{m-1}), \quad P = (\eta D + L)^{-1}$$

Algebraic system and GD

Algebraic system: $u_h = \sum u_i \varphi_i$

$$Au = f, \quad \text{where } A = ((\varphi_j', \varphi_i'))_{ij}$$

Solve it by gradient descent:

Size	4^2	16^2	64^2	256^2	1024^2
GD	56	954	14,758	223,630	$>$ 1,000,000

- The number of iterations increases dramatically for larger linear systems, leading to a poor solver.

Convergence rate of gradient descent method:

$$\|(u_h - u_h^m)\| \leq \left(1 - ch^2\right)^m \|(u_h - u_h^0)\|.$$

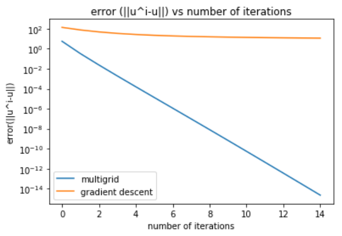


Table of Contents

Model problem and frequencies

$$\begin{cases} -u''(x) = f, & x \in (0,1), \\ u(0) = 0, & u'(1) = 0. \end{cases}$$

Consider eigenvalue problem

$$\begin{cases} -u_k''(x) = \lambda_k u_k(x), & x \in (0,1), \\ u_k(0) = 0, & u_k'(1) = 0, \end{cases}$$

We have

$$\lambda_k = (k - \frac{1}{2})^2 \pi^2, \quad u_k(x) = \sin\left((k - \frac{1}{2})(\pi x)\right), \quad k = 1, 2, 3, \cdots .$$

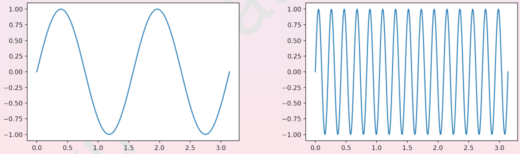


Figure: Frequencies with smaller k and larger k

Frequency bias of GD

For any SPD matrix $A \in \mathbb{R}^{n\times n}$ and vector $b \in \mathbb{R}^n$, the gradient descent method solving

$$\min_{v\in\mathbb{R}^n} I(v) \quad \text{with} \quad I(v) = \frac{1}{2}v^T Av - v^T b$$

reads as

$$v^{\ell+1} = v^{\ell} - \eta\nabla_v I(v^{\ell}), \quad \ell = 0, 1, \cdots ,$$

with initial guess v^0.
Since that $\nabla_v I(v) = Av - b$, we have

$$v^{\ell+1} = v^{\ell} - \eta(Av^{\ell} - b), \quad \ell = 0, 1, \cdots .$$

Convergence of GD with $\eta = \frac{1}{\lambda_{n,A}}$

$$v - v^{\ell} = \sum_{k=1}^{n} \alpha_k \left(1 - \frac{\lambda_{k,A}}{\lambda_{n,A}}\right)^{\ell} \xi_A^k$$

where $\xi_A^k, k = 1, 2, \cdots , n$ are the eigenvector of A.

- Fast on algebraic frequencies corresponding to large eigenvalues.
- Slow on algebraic frequencies corresponding to small eigenvalues.

H^1 fitting

Given $f \in L^{(}\Omega)$

$$J(v) = \frac{1}{2}a(v, v) - (f, v)$$

Consider to fit a target function $u(x) \in V$ by a function $u_n(x) \in V_n$.

$$a(u, v) = (u', v')_{L^2}, \quad H^1 \text{ fitting.}$$

Finite element: Piecewise linear functions

- Uniform grid $\mathcal{T}_h$

$$0 = x_0 < x_1 < \cdots < x_{N+1} = 1, \quad x_j = \frac{j}{N+1} \; (j = 0 : N+1).$$

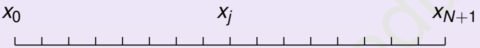


Figure: 1D uniform grid

- Linear finite element space

$$V_h = \{v_h : \; v \text{ is continuous and piecewise linear w.r.t. } \mathcal{T}_h, \; v_h(0) = 0\}.$$

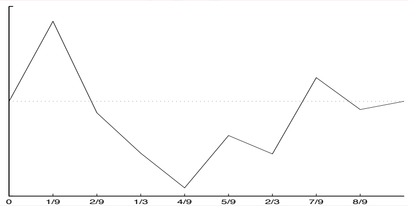
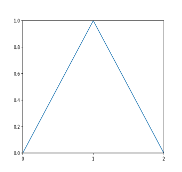


Figure: Typical finite element functions.

Two basis of the finite element space V_h

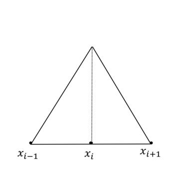


- Hat basis:

$$\varphi(x) = \begin{cases} x & x \in [0,1] \\ 2-x & x \in [1,2] \\ 0, & \text{others} \end{cases}.$$

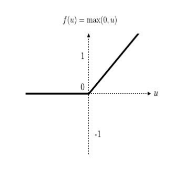
$$\varphi_i(x) = \varphi\left(\frac{x - x_{i-1}}{h}\right) = \varphi(w_h x + b_i).$$

with $w_h = \frac{1}{h}, \quad b_i = \frac{-x_{i-1}}{h}$.



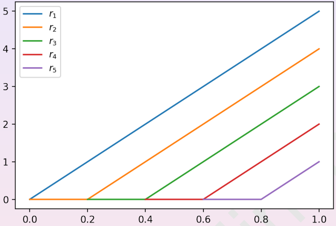
- ReLU basis: $\mathrm{ReLU}(x) = \max(0, x)$ and

$$r_i(x) = \mathrm{ReLU}\left(\frac{x - x_{i-1}}{h}\right) = \mathrm{ReLU}(w_h x + b_i)$$



- $V_h = \mathrm{span}\left\{\mathrm{ReLU}(w_h x + b_i)\right\} = \mathrm{span}\left\{\varphi(w_h x + b_i)\right\}$

Hat and ReLU bases on a uniform grid



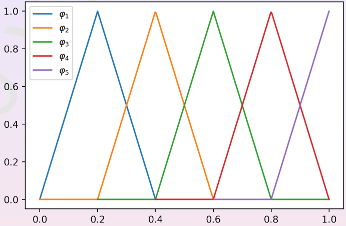


Figure: Left: ReLU bases. Right: Hat bases.

H^1-fitting

Stiffness matrix for Hat basis A_{Hat} is given by

$$A_{Hat} = \left(\int_0^1 \varphi_j'(x)\varphi_i'(x)dx \right) = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{n \times n}. \tag{8}$$

Lemma

The eigenvalues $\lambda_{k,A_{hat}}$, $1 \leq k \leq n$ and corresponding eigenvectors $\xi^k_{A_{hat}} = (\xi^k_{A_{hat},j})_{j=1}^n$, $1 \leq k \leq n$ of A_{hat} are

$$\lambda_{k,A_{hat}} = 4(n+1)^2 \sin^2 \frac{(k-\frac{1}{2})\pi}{2n+1} \approx \lambda_k,$$

$$\xi^k_{A_{hat},j} = \sin\left((k-\frac{1}{2})\pi x_j \right) \text{ with } x_j = \frac{2j}{2n+1}, 1 \leq j \leq n.$$

Frequency bias for hat basis

1. GD for stiffness matrix of Hat bases:
   - $\|\alpha - \alpha_\ell\| = \mathcal{O}\left((1 - cn^{-2})^\ell\right)$.
   - Low frequency converges slowly: $\mathcal{O}\left((1 - cn^{-2})^\ell\right)$.
   - High frequency converges fast: $\mathcal{O}(1-\delta)^\ell$ for $0 < \delta < 1$.

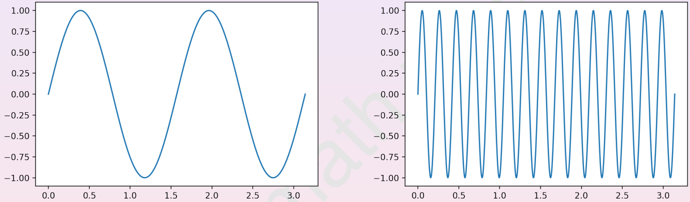


Figure: Low and high frequencies

Ref: Q. Hong, Q. Tan, J.W. Siegel, and J. Xu. On the activation function dependence of the spectral bias of neural networks. arXiv:2208:04924 (2022).

Relationship between ReLU basis and hat basis

- We have
$$\varphi(x) = 1 \cdot \mathrm{ReLU}(x) - 2 \cdot \mathrm{ReLU}(x - 1/2) + 1 \cdot \mathrm{ReLU}(x - 1). \tag{9}$$

- Let $\Psi(x) = (r_1(x), r_2(x), \cdots, r_n(x))^T$ and $\Phi(x) = (\varphi_1(x), \varphi_2(x), \cdots, \varphi_n(x))^T$. Then
$$\Phi = C\Psi, \tag{10}$$
where
$$C = \frac{1}{h^2}\begin{pmatrix} 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & \\ & & & & & 1 & \end{pmatrix}. \tag{11}$$

Spectral analysis of H^1-fitting

Stiffness matrix A_{ReLU} is given by

$$A_{ReLU} = \left(\int_0^1 r_j'(x) r_i'(x) dx\right) = h^2 \begin{pmatrix} n & n-1 & n-2 & \cdots & 1 \\ n-1 & n-1 & n-2 & \cdots & 1 \\ n-2 & n-2 & n-2 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}. \tag{12}$$

Theorem

$$A_{ReLU} = E A_{Hat}^{-1} E^{-1} \quad with \quad E = \begin{pmatrix} & & & 1 \\ & & 1 & \\ & \cdots & & \\ 1 & & & \end{pmatrix}. \tag{13}$$

The eigenvalues $\lambda_{k,A_{ReLU}}$, $1 \le k \le n$ and the corresponding eigenvectors $\xi^k_{A_{ReLU}}$, $1 \le k \le n$ of A_{ReLU} are as follows:

$$\lambda_{k,A_{ReLU}} = \lambda^{-1}_{n+1-k,A_{Hat}}, \quad \xi^k_{A_{ReLU}} = E\xi^{n+1-k}_{A_{Hat}}. \tag{14}$$

Spectral analysis of H^1-fitting

Proof:

By direct computation, we have

$$A_{ReLU} = h^2 A_1, \quad \text{with} \quad A_1 = \begin{pmatrix} n & n-1 & n-2 & \cdots & 1 \\ n-1 & n-1 & n-2 & \cdots & 1 \\ n-2 & n-2 & n-2 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix} \tag{15}$$

and

$$A_1^{-1} = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}. \tag{16}$$

By inspection, we have

$$\frac{1}{h^2} A_1^{-1} = \begin{pmatrix} & & 1 \\ & \ldots & 1 & \\ 1 & & \end{pmatrix} A_{hat} \begin{pmatrix} & & 1 \\ & \ldots & 1 & \\ 1 & & \end{pmatrix}. \tag{17}$$

Spectral analysis of H^1-fitting: eigenvectors

Theorem

Let $e_k(x) = \xi^k_{A_{ReLU}} \cdot \Psi(x) = \sum_{i=1}^{n} \xi^k_{A_{ReLU},i} r_i(x)$, *then we have*

$$e_k(x_j) = \sin\frac{\pi t_k}{2} + \sin\left((n-k+\frac{1}{2})\pi t_j - \frac{\pi t_k}{2}\right) \quad \textit{and} \quad t_j = \frac{2j}{2n+1}.$$

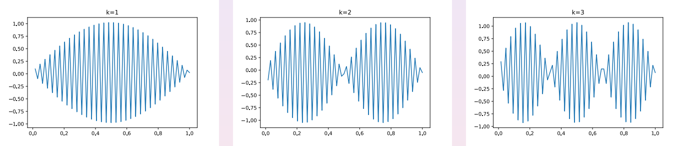


Figure: Functions: $e_1(x)$, $e_2(x)$ and $e_3(x)$.

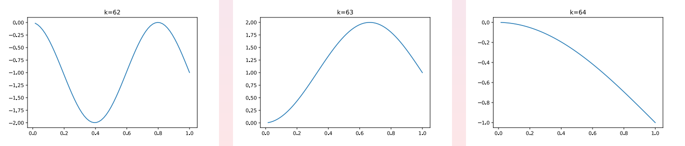


Figure: Functions: $e_{62}(x)$, $e_{63}(x)$ and $e_{64}(x)$.

Frequency bias for ReLU basis

1. GD for the stiffness matrix of ReLU basis:
   - $\|\alpha - \alpha_\ell\| = \mathcal{O}\left((1 - cn^{-2})^\ell\right)$.
   - Low frequency converges fast: $\mathcal{O}(1-\delta)^\ell$ for $0 < \delta < 1$..
   - High frequency converges slowly: $\mathcal{O}\left((1 - cn^{-2})^\ell\right)$.

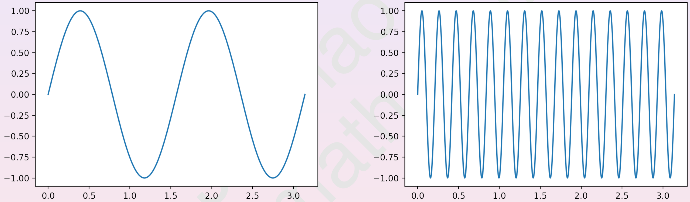


Figure: Low and high frequencies

Ref: Q. Hong, Q. Tan, J.W. Siegel, and J. Xu. On the activation function dependence of the spectral bias of neural networks. arXiv:2208:04924 (2022).

GD for H^1-fitting

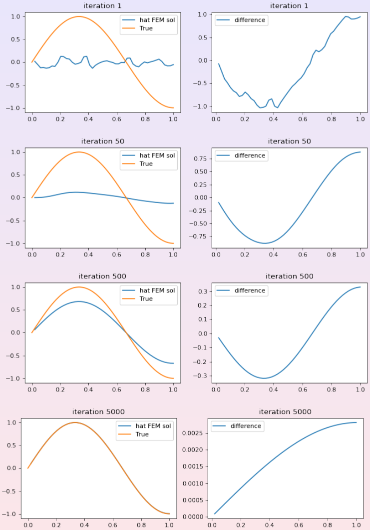


Figure: Results of Hat basis.

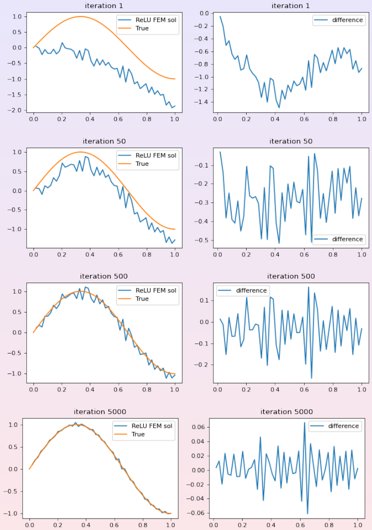
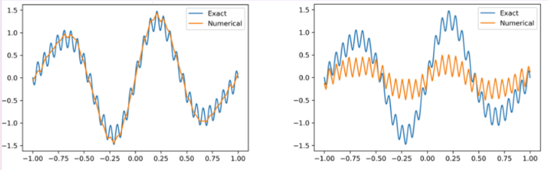


Figure: Results of ReLU basis.

Frequency bias for training neural network

- A special case of neural network functions: linear problems
- The frequency principle is still true for nonlinear problems with neural network functions.



Poisson equation. Left: ReLU activation. Right: Hat activation.

Activation dependence of training neural network

ReLU neural networks

- Prioritize learning low frequency modes in H^1 fitting
- Prioritize learning low frequency modes in L^2 fitting
- Training loss decreases slowly in L^2 fitting due to the frequency bias

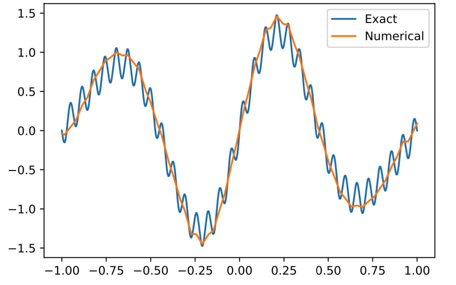
Hat neural networks

- Prioritize learning the high frequency modes in H^1 fitting
- Learn both the low frequency and high frequency modes in L^2 fitting
- Training loss decreases very fast in L^2 fitting since there is no frequency bias

- Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y. & Courville, A. (2019), Xu, Z. (2018), Cai, W. & Xu, Z. (2019), Xu, Z., Zhang, Y., Luo, T., Xiao, Y. & Ma, Z (2019), Hong, Q., Seigel, J., Tan, Q., & Xu, J. (2022).

"Convergence" of SGD or Adam Algorithms for NN-based PDE Solver

- SGD and Adam converge rather quickly for low frequency, and hence capture the “profile" of physical solutions reasonably well.



- This provides a theoretical explanation of the success of methods such as PINN.

"Non-convergence" of SGD or Adam Algorithms for NN-based PDE Solver

$$u_n = \arg\min_{v_n \in \Sigma_n^{ReLU}} J(v_n). \tag{18}$$

We have proved that one can NOT use SGD or Adam to numerically find $\tilde{u}_n \approx u_n$ such that

$$\|u - \tilde{u}_n\| \leq cn^{-\alpha} \tag{19}$$

for any $\alpha > 0$ for large n.

- H^1-fitting by ReLU NN:

  $$1 - cn^{-2}$$

  Taking $n = 10^6$: how many iterations do we need such that

  $$(1 - 10^{-12})^k \leq 10^{-7} \tag{20}$$

  - $k \geq 1.61x10^{25}$
  - 32 years for the fastest computer in the world (Frontier, 1.1 EFLOPS)

New training algorithms are required to achieve sufficiently good accuracy!

- *Greedy training algoritm*

Table of Contents

GD for a nearly singular system

Consider: $A_\epsilon u = g$ $(A_\epsilon = A_0 + \epsilon I)$

$$A_0 = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}, \quad g = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \in R(A_0), \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in N(A_0).$$

Note that $\sigma(A_0) = \{3, 1, 0\}$. Apply scaled gradient descent method with $\|A_\epsilon u^k - g\| \leq 10^{-8}$:

ϵ	# of iter $= m$
1.	37
10^{-1}	236
10^{-2}	1,918
10^{-3}	16,115
10^{-4}	130,168
0. [singular case]	20

Iterative method usually is OK for singular system, but subtle for nearly singular system!

Ref for semi-definite case: Keller 1965; Lee, Wu, Xu and Zikatanov 2007

Remedy: Expanded system (Over-parametrization)

Write $u \in \mathbb{R}^3 = u_1 e_1 + u_2 e_2 + u_3 e_3$ as

$$u = \underset{\sim}{u}_1 e_1 + \underset{\sim}{u}_2 e_2 + \underset{\sim}{u}_3 e_3 + \underset{\sim}{u}_4 p = P\underset{\sim}{u}$$

where

$$P = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad p = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in \ker(A_0).$$

Namely, we consider the coarse level with “lowest” frequency $p \in \ker(A_0)$.

The equation $A_\epsilon u = g$ becomes

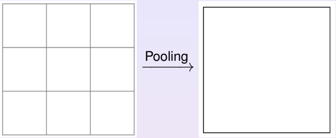
$$A_\epsilon P\underset{\sim}{u} = g \Longleftrightarrow (P^T A_\epsilon P)\underset{\sim}{u} = P^T g,$$

leading to a semi-definite system:

$$\begin{pmatrix} 1+\epsilon & -1 & 0 & \epsilon \\ -1 & 2+\epsilon & -1 & \epsilon \\ 0 & -1 & 1+\epsilon & \epsilon \\ \epsilon & \epsilon & \epsilon & 3\epsilon \end{pmatrix} \underset{\sim}{u} = \begin{pmatrix} -1 \\ -1 \\ 2 \\ 0 \end{pmatrix}.$$

# GD with $\eta = 0.7$		
ϵ	original	normalized expanded
1.	37	13
10^{-1}	236	14
10^{-2}	1,918	14
10^{-3}	16,115	16
10^{-4}	130,168	16
10^{-5}	$>$ 1,000,000	16
10^{-9}	$>$ 1,000,000	15
10^{-10}	21	15
0.	20	

Over-parametrization $\Longleftrightarrow$ Two-level methods



$$V_1 = \mathbb{R}^3 \xrightarrow[p^T]{\text{Pooling}} V_2 = \mathbb{R}$$

1. Initialization of inputs

$$A_1 = A_\epsilon, \quad g_1 \leftarrow g, \quad u_1 \leftarrow \text{random}.$$

2. Iterate:
   1. One step of GD method on V_1

   $$u_1 \leftarrow u_1 + \eta(g_1 - A_1 u_1).$$

   2. Consider $A_1 e_1 = r_1 \equiv g_1 - A_1 u_1$ and "pool" it to V_2 and solve it:

   $$A_2 u_2 = g_2, \quad u_2 = A_2^{-1} g_2$$

   with

   $$A_2 = p^T A_1 p = 3\epsilon, \quad g_2 = p^T r_1$$

   3. update $u_1 \leftarrow u_1 + p u_2$.

Multilevel method: over-parameterization using multilevel frame

Multilevel frame over-parameterization $\Longleftrightarrow$ Multigrid

$$V_J \subset V_{J-1} \subset V_{J-2} \subset \ldots \subset V_1 \equiv V.$$

Frame:

$$\{\phi_{k,i} : i = 1 : n_k, k = 1 : J\}$$

Frame expansion (not unique):

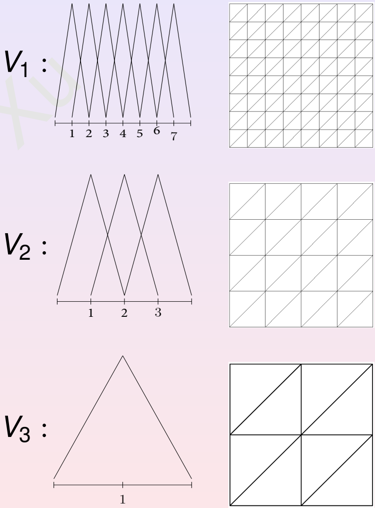
$$u_h = \sum_{k=1}^{J} \sum_{x_{k,i} \in N_k} \mu_{i,k} \phi_{k,i}.$$

Expanded system

$$\underset{\sim}{A}\underset{\sim}{\mu} = \underset{\sim}{b}$$

where $\underset{\sim}{A}$ is the frame stiffness matrix

$$\underset{\sim}{A} = \Big((\phi_{k,i}, \phi_{l,j})_A \Big) \in R^{N \times N}, \quad N = \sum_{k=1}^{J} n_k$$



An equivalent formulation of multigrid

Smoothing and restriction

- For $k = 1 : J$
  - For $i = 1 : n_k$

$$u_k \leftarrow u_k + S_k * (g_k - A_k * u_k).$$

  - Form restricted residual and set initial guess:

$$u_{k+1,0} \leftarrow \Pi_k^{k+1} u_k,$$
$$g_{k+1} \leftarrow R_k *_2 (g_k - A_k * u_k) + A_{k+1} * u_{k+1}^0.$$

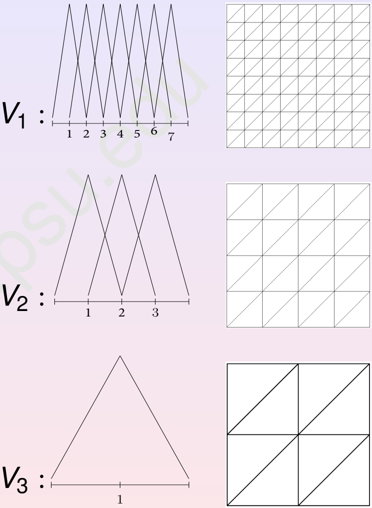
Prolongation with post-smoothing

- For $k = 1 : J - 1$

$$u_k \leftarrow u_k + R_k *_2^\top (u_{k+1} - u_{k+1}^0).$$

  - For $i = 1 : n'_k$

$$u_k \leftarrow u_k + S'_k * (g_k - A_k * u_k)$$

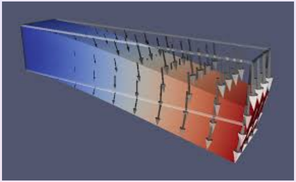


$$\phi_{3,1} = \frac{1}{2}\phi_{2,1} + \phi_{2,2} + \frac{1}{2}\phi_{2,3}$$

2D linear system on a uniform grid

- Model Problem:

$$-\Delta u := -(u_{xx} + u_{yy}) = g, \text{ in } \Omega,$$
$$u = 0 \text{ on } \partial\Omega, \quad \Omega = (0,1)^2.$$



- Discrete case:

$$4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = g_{i,j}, \tag{21}$$

with

$$A * u = g, \quad \text{for} \quad A = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}, \tag{22}$$

GD for the over-parameterized multilevel system

Original system in terms of a basis

$$Au = g, \quad A = ((\nabla\phi_j, \nabla\phi_i)) \in \mathbb{R}^{n_1 \times n_1}$$

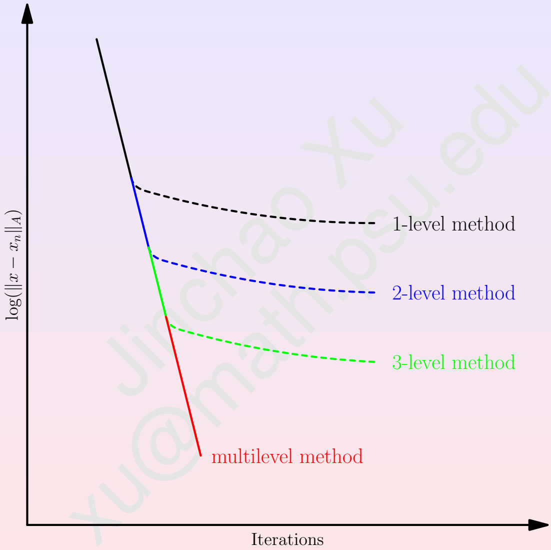
Expanded system in terms of a multilevel frame (over-parameterization):

$$\underset{\sim}{A}\underset{\sim}{u} = \underset{\sim}{g}, \quad A = ((\nabla\phi_{\ell,j}, \nabla\phi_{k,i})) \in R^{N\times N}, \quad N = \sum_{k=1}^{J} n_k$$

Solve $\underset{\sim}{A}$ by Gradient Descent:

Size	GD for A	GD for $\underset{\sim}{A}$
4^2	56	16
16^2	954	21
64^2	14,758	26
256^2	223,630	26
1024^2	>1,000,000	26

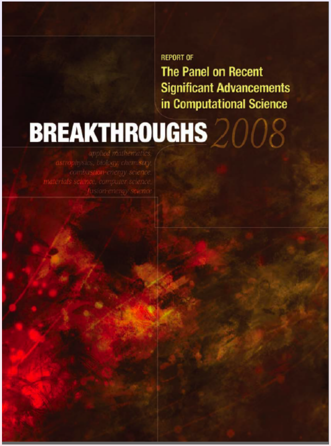
Performance of multigrid:



A success story: HX preconditioner

(Hiptmair and Xu 2005, 2007, Xu 2014)

A DOE report to the U. S. Congress



21

Novel Solver Enables Scalable Electromagnetic Simulations

ABSTRACT: A team based at Lawrence Livermore National Laboratory has developed the first provably scalable solver code for Maxwell's equations, a set of partial differential equations that are fundamental to numerous areas of physics and engineering. This new software technology enables researchers to solve larger computational problems with greater accuracy.

AMS computation of a bifilar helical coil problem used in pulsed-power experiments. Image courtesy R. N. Rieben, LLNL.

Large-scale electromagnetic simulations are often bottlenecked by slow linear solvers. In simulations conducted at LLNL, a newly developed algorithm, known as the auxiliary-space Maxwell solver (AMS), outperforms earlier solution techniques by as much as 25 times, giving researchers a significant advantage in the future as the questions they seek to answer inevitably become larger and more complex.

The solver's capability is the result of its scalability. Specifically, AMS exhibits "weak" parallel scalability, meaning that the solution time is constant as the problem size and processor workload simultaneously increase. The new algorithm is able to handle complex geometries and problems with large jumps in the material coefficients. In contrast some of the old solvers take more time and produce less accurate results when faced with systems that consist of materials of widely different electromagnetic properties, which are common in engineering.

AMS works by reducing the original problem to a series of equations that can be individually handled using classical techniques. A major advantage of this approach is that its performance is backed by a solid theoretical framework. Thus, AMS is a perfect example of how fundamental mathematical research can lead to important software advances in HPC. In this effort Panayot Vassilevski and Tzanio Kolev of LLNL collaborated with Jinchao Xu of Penn State University and Ralf Hiptmair of ETH, Zurich.

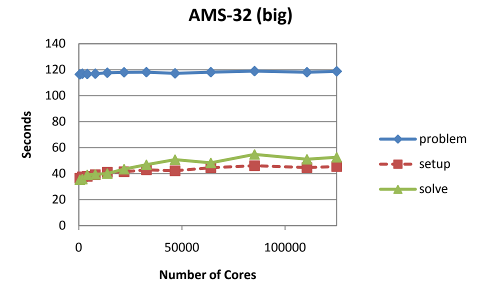
Electromagnetic simulations have a wide range of physical and engineering applications such as in the development of semiconductor chips, stealth aircraft, and electrical generators. As the ability of supercomputers to tackle ever bigger problems grows, researchers need to be able to efficiently take advantage of this newfound computing power.

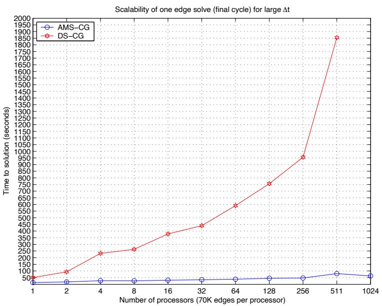
The AMS solver does just that, solving ever more complex simulations with greater accuracy. It gives researchers an edge by taming solution time, which enables a greater number of simulations,

"AMS is a perfect example of how fundamental mathematical research can lead to important software advances in high-performance computing."

One application: LLNL

Scalability of HX preconditioner to 125,000 cores





Left: Auxiliary-space Maxwell Solver. Total problem size is 12 billion.

Right: Scalability (70K edge unknowns per processor)

Ref: A. Baker, R. Falgout, T. Kolev, and U. Yang 2012

Comment and Questions

- Multigrid is powerful.
- Can the power of multigrid be transformed to CNN?
  - better structure CNN with fewer weights?
  - faster training algorithms?

Table of Contents

Space decomposition and subspace correction

- V: Hilbert space, $A\colon V \to V'$: linear operator, $f \in V'$. Find $u \in V$ such that

$$Au = f.$$

- Space decomposition: $V = \sum_i V_i = \sum_i I_i V_i$:

$$u = \sum_{i=1}^{J} u_i = \sum_{i=1}^{J} I_i u_i.$$

- Subspace solvers: $R_i : V_i' \mapsto V_i$ with

$$R_i \approx A_i^{-1}, \quad (A_i u_i, v_i) = (A u_i, v_i), u_i, v_i \in V_i$$

- Parallel subspace correction:

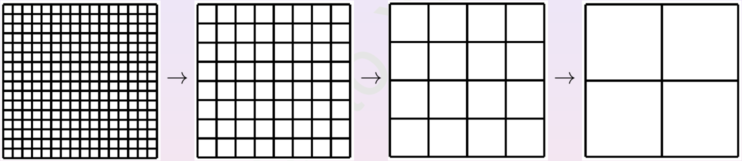
$$u \leftarrow u + B(f - Au), \qquad B = \sum_{i=1}^{J} I_i R_i I_i^T.$$

- Successive subspace correction (SSC): $\quad u \leftarrow u + I_i R_i I_i^T (f - Au),$ for $i = 1 : J$

Xu, J. (1992).

Examples

- Jacobi and block Jacobi methods are parallel subspace corection methods.
- Gauss–Seidel and block Gauss–Seidel methods are successive subspace correction methods.
- Multigrid methods:



V_1 V_2 V_3 V_4

$$V = \sum_{k=1}^{J} V_k = \sum_{k=1}^{J} \sum_{x_{k,i} \in N_k} \operatorname{span}\{\phi_{k,i}\}$$

  - Successive subspace correction $\rightarrow$ multigrid with Gauss–Seidel smoothers
  - Parallel subspace correction $\rightarrow$ BPX preconditioner

Bramble, J.H., Pasciak, J.E., and Xu, J. (1990).

Space decomposition and expanded system

- Space decomposition: $V = \sum_i V_i = \sum_i I_i V_i$:

$$u = \sum_{i=1}^{J} u_i = \sum_{i=1}^{J} I_i u_i = \Pi \underset{\sim}{u}$$

where

$$\Pi = (I_1, \ldots, I_J), \quad \underset{\sim}{u} = (u_1, \ldots u_J)^T$$

- Expanded system:

$$A\Pi\underset{\sim}{u} = Au = f \Rightarrow \Pi^T A\Pi\underset{\sim}{u} = \Pi^T f$$

- Block Jacobi and Gauss-Seidel can be applied.

Connection with Block Jacobi and Gauss-Seidel

- PSC $\Leftrightarrow$ Block Jacobi
- SSC $\Leftrightarrow$ Block Gauss-Seidel

PSC and SSC in the view from expanded system

Theorem

Iterative methods for $\underset{\sim}{A}\underset{\sim}{u} = \underset{\sim}{f}$:

$$\underset{\sim}{u}^m = \underset{\sim}{u}^{m-1} + \underset{\sim}{B}(\underset{\sim}{f} - \underset{\sim}{A}\underset{\sim}{u}^{m-1}), \quad m = 1, 2, \ldots$$

- *PSC for $Au = f \Leftrightarrow$ modified Jacobi: $\underset{\sim}{B} = \underset{\sim}{R} \approx \underset{\sim}{D}^{-1}$*
- *SSC for $Au = f \Leftrightarrow$ modified G-S: $\underset{\sim}{B} = (\underset{\sim}{R}^{-1} + \underset{\sim}{L})^{-1}$.*

Some history:

- X. 1992: DD, MG, Jacobi and GS $\Rightarrow$ PSC or SSC
- Griebel 1994: MG $\Leftrightarrow$ GS for expanded matrix in terms of multilevel nodal basis
- L. Chen 2011: PSC (SSC) $\Leftrightarrow$ Jacobi and GS for expanded matrix (as stated above)

Theory: XZ-identity

Sharp convergence theory for subspace correction methods

$$u - u^n = \prod_{i=1}^{J}(I - T_i)(u - u^{n-1}), \quad T_i = R_i A_i P_i.$$

Theorem (Xu and Zikatanov (2002, J. AMS, 2008))

The MSC is convergent if each subspace solver is convergent:

$$\|\prod_{i=1}^{J}(I - T_i)\|^2 = 1 - \frac{1}{K}, \quad K = \sup_{\|v\|=1} \inf_{\sum_i v_i = v} \sum_{i=1}^{J} \|v_i + T_i^* \sum_{j=i+1}^{J} v_j\|^2_{\bar{R}_i^{-1}}$$

Special case ($T_i = P_i$)

$$\|\prod_{i=1}^{J}(I - P_i)\|^2 = 1 - \left(\sup_{\|v\|=1} \inf_{\sum_i v_i = v} \sum_{i=1}^{J} \|P_i \sum_{j=i}^{J} v_i\|^2\right)^{-1}$$

Convergence theory of multigrid methods

Using the XZ identity, we can obtain a uniform convergence rate of the multigrid method.

Corollary (Uniform convergence of multigrid)

The convergence rate of the multigrid method for the finite element method

$$a(u, v) = f(v), \quad \forall v \in V_h$$

has a bound independent of the mesh size h.

Convex optimization

- V: Banach space, $L\colon V \to \overline{\mathbb{R}}$: convex function. Find $u \in V$ such that

$$\min_{u\in V} L(u).$$

- In many applications in machine learning, L is of the form

$$L(u) = \frac{1}{N}\sum_{i=1}^{N} f_i(u).$$

- Gradient descent type methods
  - Full (batch) gradient descent

$$u_{t+1} = u_t - \eta_t \nabla\left(\frac{1}{N}\sum_{i=1}^{N} f_i(u_t)\right).$$

  - Stochastic gradient descent (SGD)

$$u_{t+1} = u_t - \eta_t \nabla f_{i_t}(u_t),$$

where $\mathrm{Pr}(i_t = k) = \frac{1}{N}$.

Convergence of SGD

Theorem

Assume that each $f_i(u)$ is λ-strongly convex and $\|\nabla f_i(u)\| \le M$ for some $M > 0$. If we take

$$\eta_t = \frac{a}{\lambda(t+1)}$$

with sufficiently large a such that

$$\|u_0 - u^*\|^2 \le \frac{a^2 M^2}{(a-1)\lambda^2} \tag{23}$$

then

$$\mathbb{E} e_t^2 \le \frac{a^2 M^2}{(a-1)\lambda^2(t+1)}, \quad t \ge 1, \tag{24}$$

where $e_t = \|u_t - u^*\|$.

Convergence of SGD

Proof: Note that

$$\begin{aligned}
\mathbb{E}(\nabla f_{i_t}(u_t)\cdot(u_t-u^*)) &= \mathbb{E}_{i_1 i_2\cdots i_t}(\nabla f_{i_t}(u_t)\cdot(u_t-u^*)) \\
&= \mathbb{E}_{i_1 i_2\cdots i_{t-1}}\frac{1}{N}\sum_{i=1}^{N}\nabla f_i(u_t)\cdot(u_t-u^*) \\
&= \mathbb{E}_{i_1 i_2\cdots i_{t-1}}\nabla f(u_t)\cdot(u_t-u^*) \\
&= \mathbb{E}\nabla f(u_t)\cdot(u_t-u^*),
\end{aligned} \tag{25}$$

and $\mathbb{E}\|\nabla f_{i_t}(u_t)\|^2 \le \mathbb{E}M^2 = M^2$.
The L^2 error of SGD can be written as

$$\begin{aligned}
\mathbb{E}\|u_{t+1}-u^*\|^2 &\le \mathbb{E}\|u_t-\eta_t\nabla f_{i_t}(u_t)-u^*\|^2 \\
&= \mathbb{E}\|u_t-u^*\|^2 - 2\eta_t\mathbb{E}(\nabla f_{i_t}(u_t)\cdot(u_t-u^*)) + \eta_t^2\mathbb{E}\|\nabla f_{i_t}(u_t)\|^2 \\
&\le \mathbb{E}\|u_t-u^*\|^2 - 2\eta_t\mathbb{E}(\nabla f(u_t)\cdot(u_t-u^*)) + \eta_t^2 M^2.
\end{aligned} \tag{26}$$

By the definition of λ-strongly convex

$$\nabla f(u_t)\cdot(u^*-u_t)+\frac{\lambda}{2}\|u^*-u_t\|^2 \le f(u_t)-f(u^*)+\nabla f(u_t)\cdot(u^*-u_t)+\frac{\lambda}{2}\|u^*-u_t\|^2 \le 0. \tag{27}$$

Convergence of SGD

Thus,

$$\begin{aligned}\mathbb{E}\|u_{t+1}-u^*\|^2 &\le \mathbb{E}\|u_t-u^*\|^2-\eta_t\lambda\mathbb{E}\|u_t-u^*\|^2+\eta_t^2M^2\\ &=(1-\eta_t\lambda)\mathbb{E}\|u_t-u^*\|^2+\eta_t^2M^2\\ &=(1-\frac{a}{t+1})\mathbb{E}\|u_t-u^*\|^2+\frac{a^2M^2}{\lambda^2(t+1)^2}\end{aligned} \tag{28}$$

When $t=0$, we have, based on the assumption

$$\mathbb{E}e_0^2=\|u_0-u^*\|^2\le\frac{a^2M^2}{(a-1)\lambda}, \tag{29}$$

We complete the proof using mathematical induction. Suppose(24) holds for t, since $\frac{t}{(t+1)^2}\le\frac{1}{t+2}$,

$$\begin{aligned}\mathbb{E}e_{t+1}^2 &\le (1-\frac{a}{t+1})\mathbb{E}\|u_t-u^*\|^2+\frac{a^2M^2}{\lambda^2(t+1)^2}\\ &\le(1-\frac{a}{t+1})\frac{a^2M^2}{(a-1)\lambda^2(t+1)}+\frac{a^2M^2}{\lambda^2(t+1)^2}\\ &\le\frac{a^2M^2}{(a-1)\lambda^2}\frac{1}{(t+1)^2}(t+1-a+a-1)\\ &=\frac{a^2M^2}{(a-1)\lambda^2}\frac{t}{(t+1)^2}\le\frac{a^2M^2}{(a-1)\lambda^2(t+2)}.\end{aligned} \tag{30}$$

Subspace correction methods for convex optimization

- V: Banach space, $L: V \to \overline{\mathbb{R}}$: convex function. Find $u \in V$ such that

$$\min_{u \in V} L(u).$$

- Space decomposition $V = \sum_{i=1}^{J} V_i$, $u = \sum_{i=1}^{J} u_i$
- Local corrections in subspaces: Find $w_i \in V_i$ such that

$$\min_{w_i \in V_i} L(u + w_i)$$

- Successive subspace correction (SSC):

$$u \leftarrow u + w_i, \text{ for } i = 1 : J$$

- Parallel subspace correction (PSC):

$$u \leftarrow u + \tau \sum_{i=1}^{J} w_i$$

Ref. Tai, X.-C. and Xu, J. (2002), Park, J. (2020)

Convergence theory

- L is M-smooth, i.e.,

$$L(u) \le L(v) + \langle L'(v), u - v\rangle + \frac{M}{2}\|u - v\|^2, \quad \forall u, v \in V.$$

- L is μ-strongly convex, i.e.,

$$L(u) \ge L(v) + \langle L'(v), u - v\rangle + \frac{\mu}{2}\|u - v\|^2, \quad \forall u, v \in V.$$

Theorem (Tai and Xu (2002), Park (2020))

The MSC for convex optimization is convergent. Morever, we have

$$\frac{L(u^n) - L(u)}{L(u^{n-1}) - L(u)} \le 1 - \frac{1}{K},$$

where

$$K \approx \mu^{-1} \sup_{\|w\|=1} \inf_{w=\sum_{i=1}^J w_i} \sum_{i=1}^J \|w_i\|^2.$$

An application: Federated learning

We consider the following N-client training model:

$$\min_{\theta\in\Omega}\left\{L(\theta) := \frac{1}{N}\sum_{i=1}^{N} f_i(\theta)\right\}$$

- N: number of clients (devices)
- f_i: loss on local data stored on the client i

Conventionial training (GD)

$$\theta \leftarrow \theta - \eta\nabla L(\theta)$$

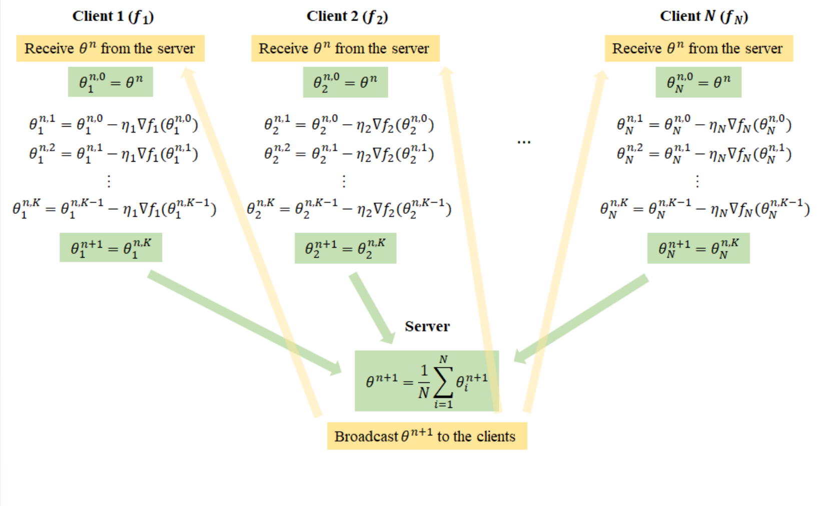
Federated learning (FL)
Each client performs local training (several GD steps) using its local function f_i, and the results are averaged in the server.

FedAvg: McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B.A.y. (2017),
Scaffold: Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A.T. (2020),
Scaffnew: Mishchenko, K., Malinovsky, G., Stich, S., and Richtarik, P. (2022),

DualFL: Park, J. and Xu, J. (2023).

An application: Federated learning

Federated learning (FL)



Question: By modifying local trainings and global communications, can we design federated learning algorithms with fewer communication costs?

Federated Learning ↔ Parallel Subspace Correction

Federated learning problem

$$\min_{\theta\in\Omega}\left\{L(\theta) := \frac{1}{N}\sum_{i=1}^{N} f_i(\theta)\right\}$$

Fenchel–Rockafellar duality

$$\theta = -\frac{1}{N\nu}\sum_{i=1}^{N}\xi_i, \quad \xi_i = \nabla g_i(\theta)$$

Dual problem:

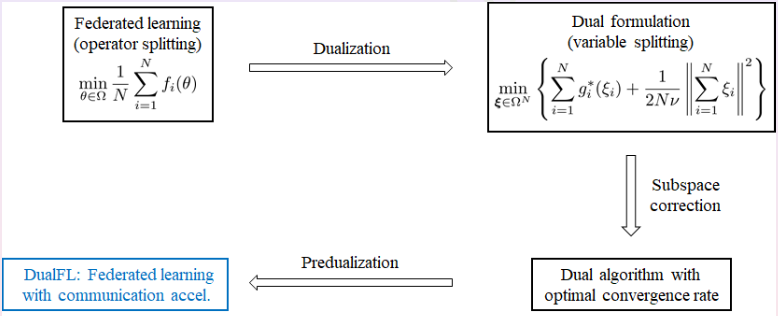
$$\min_{\boldsymbol{\xi}\in\Omega^N}\left\{L_d(\boldsymbol{\xi}) := \sum_{i=1}^{N} g_i^*(\xi_i) + \frac{1}{2N\nu}\left\|\sum_{i=1}^{N}\xi_i\right\|^2\right\}.$$

- $\nu \in (0, \mu]$
- $g_i = f_i - \frac{\mu}{2}\|\cdot\|^2$
- $g_i^* : \Omega \to \overline{\mathbb{R}}$: convex conjugate of g defined by

$$g_i^*(\phi) = \sup_{\theta\in\Omega}\{\langle\phi,\theta\rangle - g_i(\theta)\}$$

Federated Learning $\leftrightarrow$ Parallel Subspace Correction

By establishing a duality relation between federated learning and parallel subspace correction methods, we design a new federated learning algorithm with optimal communication complexity.



- Park, J., & Xu, J. (2023).

Communication efficiency

Theorem (J. Park and J. Xu, 2023)

In DualFL, the number of communication rounds M to obtain an ϵ-accurate solution satisfies

$$M = \begin{cases} \mathcal{O}\left(\sqrt{\frac{L}{\mu}} \log \frac{1}{\epsilon}\right), & \text{if each } f_i \text{ is } \mu\text{-strongly convex and } L\text{-smooth,} \\ \mathcal{O}\left(\frac{1}{\sqrt{\epsilon}}\right), & \text{if each } f_i \text{ is } \mu\text{-strongly convex,} \\ \mathcal{O}\left(\frac{1}{\sqrt{\epsilon}} \log \frac{1}{\epsilon}\right), & \text{if each } f_i \text{ is convex and } L\text{-smooth.} \end{cases}$$

- Xu, J. (1992) Tai, X.-C & Xu, J. (2002), Park, J., & Xu, J. (2023), Park, J. (2020)