Section 09. Robust Iterative Methods

A Simple Model Problem

- A representative example is the second-order elliptic problem with anisotropic coefficient

  $$-\epsilon u_{xx} - u_{yy} = f(x,y), \quad \forall\,(x,y) \in \Omega,$$

  where $\epsilon > 0$ is usually small.

- If we apply the standard finite difference discretization on the uniform $n \times n$ tensor-product grid, then

  $$A_\epsilon = I \otimes A_{1,\epsilon} + C \otimes I, \quad \text{with } A_{1,\epsilon} = \operatorname{tridiag}(-\epsilon, 2+2\epsilon, -\epsilon),\ C = \operatorname{tridiag}(-1, 0, -1).$$

- The eigenvalues of A are given

  $$\lambda_{i,j}(A_\epsilon) = 2(1+\epsilon) - 2\epsilon \cos\frac{i\pi}{n+1} - 2\cos\frac{j\pi}{n+1} = 4\epsilon \sin^2\frac{i\pi}{2(n+1)} + 4\sin^2\frac{j\pi}{2(n+1)},$$

  with eigenvectors

  $$\vec{\xi}_{i,j} = \left(\sin\frac{ki\pi}{n+1} \sin\frac{lj\pi}{n+1}\right)_{k,l=1,\ldots,n}.$$

- If $\epsilon \ll 1$, then $\lambda_{1,1} < \lambda_{2,1} < \cdots < \lambda_{n,1} < \lambda_{1,2} < \lambda_{2,2} < \cdots$.

- Unlike the Poisson's equation, these eigenvalues are ordered in a different pattern.

Local Relaxation Method In Danger

Error smoothness is not trivial to define for problems for complex problems in general.

- Using the LFA analysis, we obtain that the error of the G-S method satisfies

$$(2+2\epsilon)e_{i,j}^{\text{new}} = \epsilon e_{i-1,j}^{\text{new}} + \epsilon e_{i+1,j}^{\text{old}} + e_{i,j-1}^{\text{new}} + e_{i,j+1}^{\text{old}}, \quad i,j=1,\ldots,n.$$

- According to the local Fourier analysis, we can obtain that

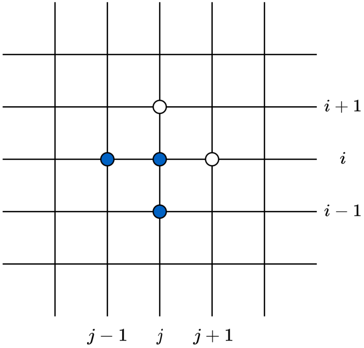
$$\lambda(\theta_1,\theta_2) := \frac{\alpha_\theta^{\text{new}}}{\alpha_\theta^{\text{old}}} = \frac{\epsilon e^{\sqrt{-1}\theta_1} + e^{\sqrt{-1}\theta_2}}{2+2\epsilon-\epsilon e^{-\sqrt{-1}\theta_1} - e^{-\sqrt{-1}\theta_2}}.$$

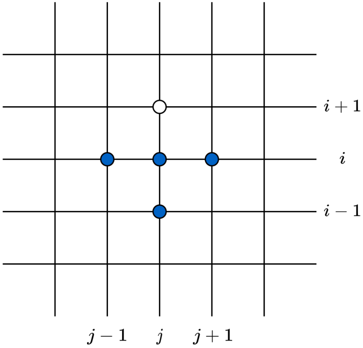
- In this case, the smoothing factor of the G-S method is

$$\bar{\rho}_{\text{GS}} = \lambda\left(\frac{\pi}{2}, \arctan\left(\frac{\epsilon(1-\bar{\rho}_{\text{GS}}^2)}{2(\epsilon+1)\bar{\rho}_{\text{GS}}^2}\right)\right) = \frac{\sqrt{5\epsilon^2-2\epsilon+1}+2}{5\epsilon+3} \longrightarrow 1, \quad \text{as } \epsilon \to 0.$$

- This observation suggests that the G-S method barely have any smoothing effect for small ϵ.

Line Smoother for Anisotropic Problems





- Standard (left) and line Gauss–Seidel (right) smoothers: Blue points have updated values and white points have old values.
- We apply the line smoother in natural ordering:

$$(2+2\epsilon)u_{i,j}^{\text{new}} = \epsilon u_{i-1,j}^{\text{new}} + \epsilon u_{i+1,j}^{\text{old}} + u_{i,j-1}^{\text{new}} + u_{i,j+1}^{\text{new}}, \quad j=1,\ldots,n, \;\; i=1,\ldots,n.$$

LFA for Line Smoother

- The error satisfies

$$(2+2\epsilon)e_{i,j}^{\text{new}} = \epsilon e_{i-1,j}^{\text{new}} + \epsilon e_{i+1,j}^{\text{old}} + e_{i,j-1}^{\text{new}} + e_{i,j+1}^{\text{new}}, \quad j = 1, \ldots, n, \;\; i = 1, \ldots, n.$$

- By LFA, we can obtain that

$$\lambda(\theta_1, \theta_2) := \frac{\alpha_\theta^{\text{new}}}{\alpha_\theta^{\text{old}}} = \frac{\epsilon e^{\sqrt{-1}\theta_1}}{2+2\epsilon-\epsilon e^{-\sqrt{-1}\theta_1} - 2e^{-\sqrt{-1}\theta_2}}.$$

- The maximal smoothing factor is $\bar{\rho}_{\text{LGS}} = \max\left\{\frac{\epsilon}{2+\epsilon}, \frac{\sqrt{5}}{5}\right\}$.
- If $0 < \epsilon \leq 1$, we always have $\bar{\rho}_{\text{LGS}} = \sqrt{5}/5 < 1$ independent of ϵ.

This example illustrates a typical problem used by researchers to evaluate the robustness of multigrid methods as well as other iterative solvers. Other examples include problems with high-contrast coefficients, heterogeneous coefficients, anisotropic meshes, etc.

Methods for Improving Performance

(1) Apply an line smoother (group all those y-variables corresponding to the same x-coordinate together)

(2) Employ y-semi-coarsening (only coarse in the y-direction)

(3) Construct operator-dependent interpolations

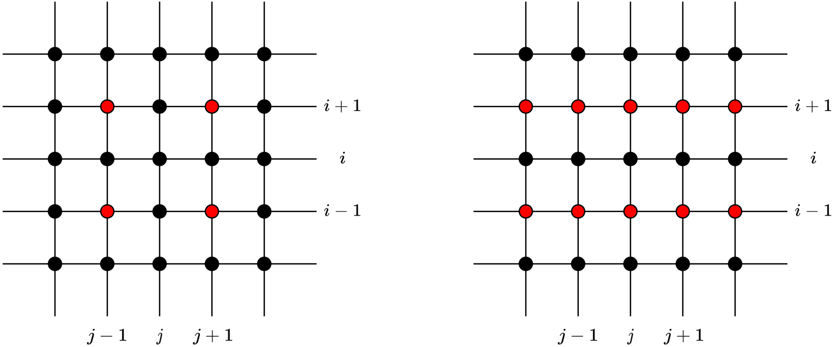


Figure: Examples of coarsening methods (Left: standard coarsening; Right: y-semi-coarsening): Red depicts coarse points and black depicts fine points.

Another Simple Model Problem

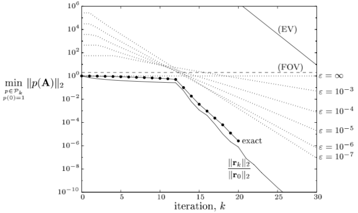
Example (2D convection-diffusion [Notay 2020])

Consider $-\nabla \cdot (\mu \nabla u + \mathbf{v} u) = 0$ on the unit square discretized by an upwinding FD on uniform meshes. Solve the discrete system by TG + weighted Jacobi smoother.

Let $\mu = 1$, $\mathbf{v} = \big(m(\alpha-1), m(\beta-1)\big)^T$, and meshsize $h = \frac{1}{m}$. Simplest setting: $m = 30$, $\alpha = 50$, $\beta = 1$, and damping $\omega = 0.5$, geometric coarsening, bilinear interpolation, and full-weighting restriction. We have the FD stencil:

$$\begin{bmatrix} & -1 & \\ -\alpha & 2+\alpha+\beta & -1 \\ & -\beta & \end{bmatrix}_h \xrightarrow{\text{symmetrize}} \begin{bmatrix} & -\frac{\beta+1}{2} & \\ -\frac{\alpha+1}{2} & 2+\alpha+\beta & -\frac{\alpha+1}{2} \\ & -\frac{\beta+1}{2} & \end{bmatrix}_h$$

non-symmetric anisotropic



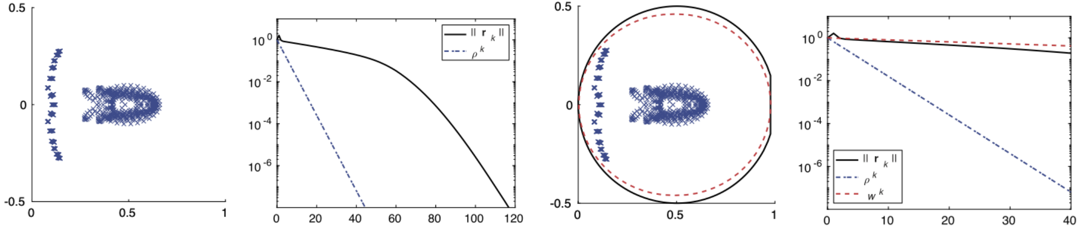
- Example: $\mu = 0.01$, $\mathbf{v} = \big(0, 1\big)^T$, $N = 13$
- Typical behavior for non-normal matrices
- Estimates: eigenvalues (EV), field of values (FOV), and pseudo-spectra (ε)
- Theoretical and numerical comparisons [Embree 1999]

Solving with Two-grid Method

- Blue crosses denote the spectrum of iteration matrix $E \implies \rho(E)$ predicts asymptotic convergence behavior; but it might be miss-leading in practice.
- Red dashed curve denotes boundary of the field of values (numerical range).
- Black solid curve is given based on the estimate for the numerical range

$$W_{P^\perp}(T) \subset \left\{ z \in \mathbb{C} \,:\, \mathrm{Re}(z) \le \rho(T_H) \ \text{ and } \ \left|\frac{1}{2} - z\right| \le \frac{1}{2} \right\},$$

which applies to the weighted Jacobi smoother [Notay 2020].



Same convergence behavior could also happen for multigrid methods!

Discretization Methods for Convection

Streamline-Upwinding Petrov-Galerkin: [Hughes, Brooks 1979; Brooks, Hughes 1982; ...]

- Equivalent to adding a perturbation term: $\sum_{E\in\mathcal{T}_h} \mu_E \int_E \left(\mathbf{v}\cdot\nabla u,\, \mathbf{v}\cdot\nabla w\right)$
- Popular in computational fluid dynamics (CFD packages, 7000+ citations)
- Introduce more nonzero entries to the linear system

Edge-Averaged Finite Element: [Xu, Zikatanov 1999; Lazarov, Zikatanov 2012]

- Idea in 1D: change of variable

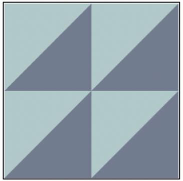
$$\begin{aligned} u' + vu = e^{-vx}(e^{vx}u)' \quad &\Longrightarrow \quad \frac{ve^{vx_i}}{e^{vx_{i+1}} - e^{vx_i}} u_i - \frac{ve^{vx_{i+1}}}{e^{vx_{i+1}} - e^{vx_i}} u_{i+1} \\ &\Longrightarrow \quad \frac{1}{h}\Big(B(vh)u_i - B(-vh)u_{i+1}\Big) \end{aligned}$$

- Can be implemented by modifying the discrete system of Poisson

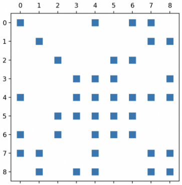
  $\Longrightarrow$ Will not expand the stencil pattern $\Longrightarrow$ At least keep sparsity

- A monotone scheme on Delaunay triangulations $\Longrightarrow$ Gives an M coefficient matrix

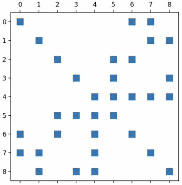
Discrete Systems of SUPG / EAFE



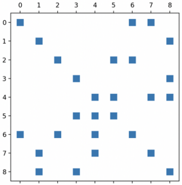
mesh



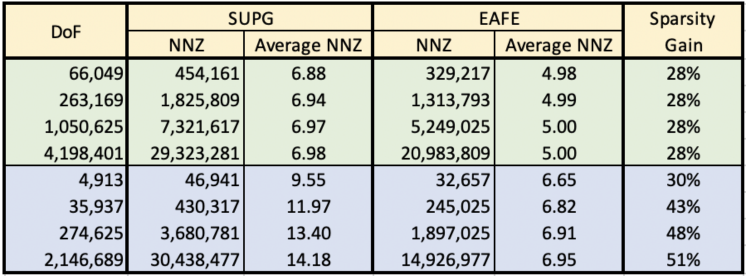
SUPG



EAFE before modification



EAFE



DoF	SUPG		EAFE		Sparsity Gain
	NNZ	Average NNZ	NNZ	Average NNZ	
66,049	454,161	6.88	329,217	4.98	28%
263,169	1,825,809	6.94	1,313,793	4.99	28%
1,050,625	7,321,617	6.97	5,249,025	5.00	28%
4,198,401	29,323,281	6.98	20,983,809	5.00	28%
4,913	46,941	9.55	32,657	6.65	30%
35,937	430,317	11.97	245,025	6.82	43%
274,625	3,680,781	13.40	1,897,025	6.91	48%
2,146,689	30,438,477	14.18	14,926,977	6.95	51%

A simple case $\mathbf{v} = (1,0)^T$, P^1 finite element [Fan, Yue, Z., Report, 2020]

AMG Preconditioned GMRES

Keep the following questions in mind throughout this lecture:

- How to choose solver parameters? For example, parameters for AMG methods ...
- Are there "best" parameters anyways? Is it practical to find them?

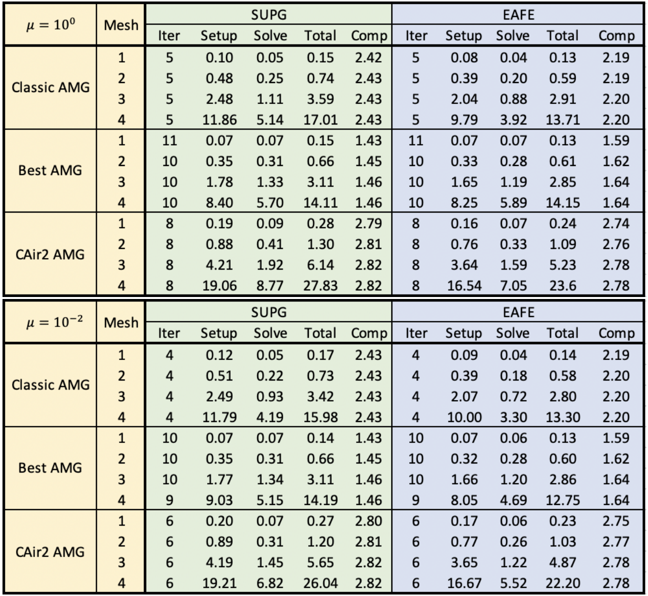
Numerical studies

- The convection-diffusion eqn (fixed convection, variable diffusion coefficient)
- Focus on AMG-GMRES: diffusion coefficient, discretization, meshsize, AMG

AMG parameters

- Tested using hypre: MaxIter = 200, tol = 1E-8, SizeOfCoarsestLevel = 100, ...
- Classical: Falgout, strong threshold 0.25, classical interpolation, GS smoother with C/F ordering, ...
- Best practices: HMIS (aggressive coarsening on the finest level only), strong threshold 0.25, distance-two interpolation, GS smoother with C/F ordering, truncate small nonzeros (keep at most 5 in each row), ...
- CAir2: Falgout, strong threshold for coarsening and restriction 0.2 and 0.05, classical interpolation, ℓAIR_2 restriction (F-F-C connections), Jacobi smoother (1F,2F), ...

Preliminary Results, Good News



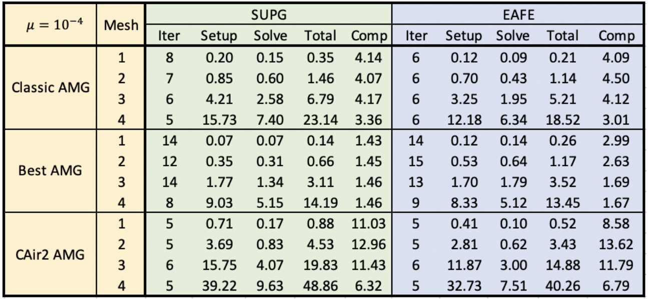
$\mu = 10^0$	Mesh	SUPG					EAFE				
		Iter	Setup	Solve	Total	Comp	Iter	Setup	Solve	Total	Comp
Classic AMG	1	5	0.10	0.05	0.15	2.42	5	0.08	0.04	0.13	2.19
	2	5	0.48	0.25	0.74	2.43	5	0.39	0.20	0.59	2.19
	3	5	2.48	1.11	3.59	2.43	5	2.04	0.88	2.91	2.20
	4	5	11.86	5.14	17.01	2.43	5	9.79	3.92	13.71	2.20
Best AMG	1	11	0.07	0.07	0.15	1.43	11	0.07	0.07	0.13	1.59
	2	10	0.35	0.31	0.66	1.45	10	0.33	0.28	0.61	1.62
	3	10	1.78	1.33	3.11	1.46	10	1.65	1.19	2.85	1.64
	4	10	8.40	5.70	14.11	1.46	10	8.25	5.89	14.15	1.64
CAir2 AMG	1	8	0.19	0.09	0.28	2.79	8	0.16	0.07	0.24	2.74
	2	8	0.88	0.41	1.30	2.81	8	0.76	0.33	1.09	2.76
	3	8	4.21	1.92	6.14	2.82	8	3.64	1.59	5.23	2.78
	4	8	19.06	8.77	27.83	2.82	8	16.54	7.05	23.6	2.78

$\mu = 10^{-2}$	Mesh	SUPG					EAFE				
		Iter	Setup	Solve	Total	Comp	Iter	Setup	Solve	Total	Comp
Classic AMG	1	4	0.12	0.05	0.17	2.43	4	0.09	0.04	0.14	2.19
	2	4	0.51	0.22	0.73	2.43	4	0.39	0.18	0.58	2.20
	3	4	2.49	0.93	3.42	2.43	4	2.07	0.72	2.80	2.20
	4	4	11.79	4.19	15.98	2.43	4	10.00	3.30	13.30	2.20
Best AMG	1	10	0.07	0.07	0.14	1.43	10	0.07	0.06	0.13	1.59
	2	10	0.35	0.31	0.66	1.45	10	0.32	0.28	0.60	1.62
	3	10	1.77	1.34	3.11	1.46	10	1.66	1.20	2.86	1.64
	4	9	9.03	5.15	14.19	1.46	9	8.05	4.69	12.75	1.64
CAir2 AMG	1	6	0.20	0.07	0.27	2.80	6	0.17	0.06	0.23	2.75
	2	6	0.89	0.31	1.20	2.81	6	0.77	0.26	1.03	2.77
	3	6	4.19	1.45	5.65	2.82	6	3.65	1.22	4.87	2.78
	4	6	19.21	6.82	26.04	2.82	6	16.67	5.52	22.20	2.78

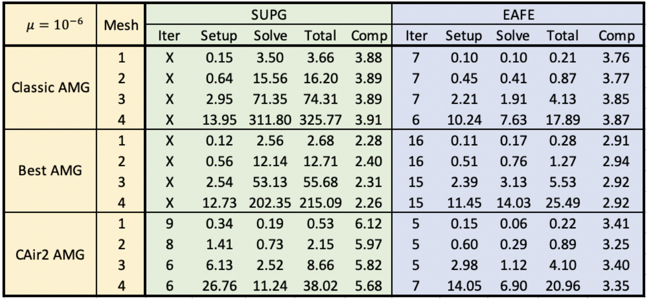
- AMGs work well and converge uniformly w.r.t. h
- Best practices AMG gives best performance
- EAFE is easier to solve, compared with SUPG
- CAir2 takes longer setup time and yields higher operator complexity

Best practices AMG is good, but not much better for EAFE discretization.

Preliminary Results, Not So Good News



$\mu = 10^{-4}$	Mesh	SUPG					EAFE				
		Iter	Setup	Solve	Total	Comp	Iter	Setup	Solve	Total	Comp
Classic AMG	1	8	0.20	0.15	0.35	4.14	6	0.12	0.09	0.21	4.09
	2	7	0.85	0.60	1.46	4.07	6	0.70	0.43	1.14	4.50
	3	6	4.21	2.58	6.79	4.17	6	3.25	1.95	5.21	4.12
	4	5	15.73	7.40	23.14	3.36	6	12.18	6.34	18.52	3.01
Best AMG	1	14	0.07	0.07	0.14	1.43	14	0.12	0.14	0.26	2.99
	2	12	0.35	0.31	0.66	1.45	15	0.53	0.64	1.17	2.63
	3	14	1.77	1.34	3.11	1.46	13	1.70	1.79	3.52	1.69
	4	8	9.03	5.15	14.19	1.46	9	8.33	5.12	13.45	1.67
CAir2 AMG	1	5	0.71	0.17	0.88	11.03	5	0.41	0.10	0.52	8.58
	2	5	3.69	0.83	4.53	12.96	5	2.81	0.62	3.43	13.62
	3	6	15.75	4.07	19.83	11.43	6	11.87	3.00	14.88	11.79
	4	5	39.22	9.63	48.86	6.32	5	32.73	7.51	40.26	6.79



$\mu = 10^{-6}$	Mesh	SUPG					EAFE				
		Iter	Setup	Solve	Total	Comp	Iter	Setup	Solve	Total	Comp
Classic AMG	1	X	0.15	3.50	3.66	3.88	7	0.10	0.10	0.21	3.76
	2	X	0.64	15.56	16.20	3.89	7	0.45	0.41	0.87	3.77
	3	X	2.95	71.35	74.31	3.89	7	2.21	1.91	4.13	3.85
	4	X	13.95	311.80	325.77	3.91	6	10.24	7.63	17.89	3.87
Best AMG	1	X	0.12	2.56	2.68	2.28	16	0.11	0.17	0.28	2.91
	2	X	0.56	12.14	12.71	2.40	16	0.51	0.76	1.27	2.94
	3	X	2.54	53.13	55.68	2.31	15	2.39	3.13	5.53	2.92
	4	X	12.73	202.35	215.09	2.26	15	11.45	14.03	25.49	2.92
CAir2 AMG	1	9	0.34	0.19	0.53	6.12	5	0.15	0.06	0.22	3.41
	2	8	1.41	0.73	2.15	5.97	5	0.60	0.29	0.89	3.25
	3	6	6.13	2.52	8.66	5.82	5	2.98	1.12	4.10	3.40
	4	6	26.76	11.24	38.02	5.68	7	14.05	6.90	20.96	3.35

- If AMGs converge, they converge uniformly
- Standard AMG does not work for SUPG when μ is small
- CAir2 is more robust, but may lead to higher complexity
- For EAFE with small μ, best practices AMG is not the best any more

Need good parameters to achieve good AMG performance!

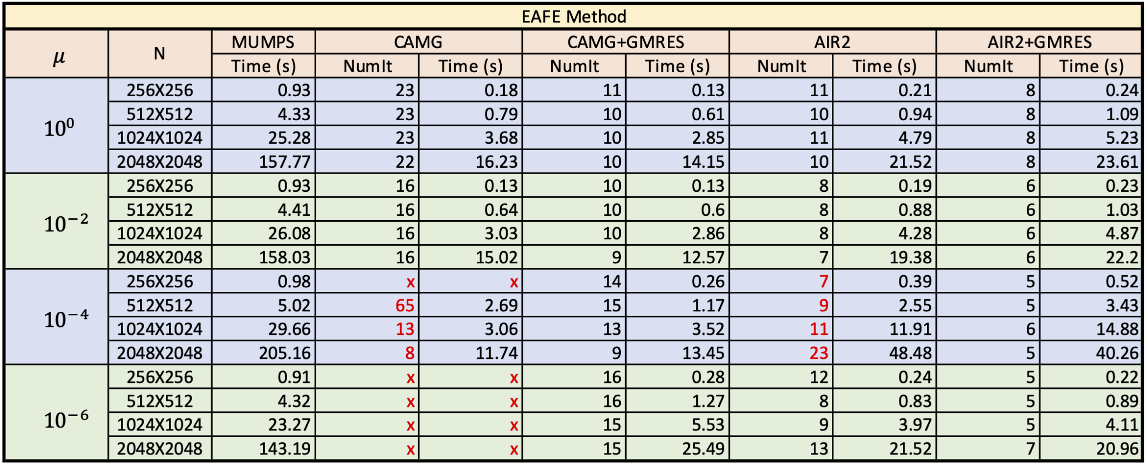
Performance of Linear Solvers for SUPG

Linear solution methods (direct solver, classical AMG, approximate ideal restriction AMG, ...)

SUPG Method										
μ	N	MUMPS	CAMG		CAMG+GMRES		AIR2		AIR2+GMRES	
		Time (s)	NumIt	Time (s)	NumIt	Time (s)	NumIt	Time (s)	NumIt	Time (s)
10^0	256X256	0.88	23	0.18	11	0.15	11	0.25	8	0.28
	512X512	5.02	23	0.89	10	0.66	10	1.14	8	1.31
	1024X1024	29.56	23	4.34	10	3.11	11	5.67	8	6.14
	2048X2048	203.17	22	17.63	10	14.11	10	25.22	8	27.83
10^{-2}	256X256	0.89	16	0.15	10	0.14	8	0.23	6	0.27
	512X512	4.85	16	0.73	10	0.66	8	1.06	6	1.21
	1024X1024	29.17	16	3.65	10	3.11	8	5.05	6	5.65
	2048X2048	203.34	15	16.48	9	14.19	7	22.57	6	26.04
10^{-4}	256X256	0.89	14	0.21	14	0.14	9	0.71	5	0.88
	512X512	5.47	13	0.85	12	0.66	48	7.46	5	4.53
	1024X1024	33.52	29	5.43	14	3.11	58	37.18	6	19.83
	2048X2048	265.11	9	12.63	8	14.19	19	54.71	5	48.86
10^{-6}	256X256	0.89	x	x	x	x	21	0.57	9	0.53
	512X512	5.18	x	x	x	x	13	1.94	8	2.15
	1024X1024	43.08	x	x	x	x	9	7.48	6	8.66
	2048X2048	371.99	x	x	x	x	8	32.55	6	38.02

Performance of Linear Solvers for EAFE

Linear solution methods (direct solver, classical AMG, approximate ideal restriction AMG, ...)



EAFE Method										
μ	N	MUMPS	CAMG		CAMG+GMRES		AIR2		AIR2+GMRES	
		Time (s)	NumIt	Time (s)	NumIt	Time (s)	NumIt	Time (s)	NumIt	Time (s)
10^0	256X256	0.93	23	0.18	11	0.13	11	0.21	8	0.24
	512X512	4.33	23	0.79	10	0.61	10	0.94	8	1.09
	1024X1024	25.28	23	3.68	10	2.85	11	4.79	8	5.23
	2048X2048	157.77	22	16.23	10	14.15	10	21.52	8	23.61
10^{-2}	256X256	0.93	16	0.13	10	0.13	8	0.19	6	0.23
	512X512	4.41	16	0.64	10	0.6	8	0.88	6	1.03
	1024X1024	26.08	16	3.03	10	2.86	8	4.28	6	4.87
	2048X2048	158.03	16	15.02	9	12.57	7	19.38	6	22.2
10^{-4}	256X256	0.98	x	x	14	0.26	7	0.39	5	0.52
	512X512	5.02	65	2.69	15	1.17	9	2.55	5	3.43
	1024X1024	29.66	13	3.06	13	3.52	11	11.91	6	14.88
	2048X2048	205.16	8	11.74	9	13.45	23	48.48	5	40.26
10^{-6}	256X256	0.91	x	x	16	0.28	12	0.24	5	0.22
	512X512	4.32	x	x	16	1.27	8	0.83	5	0.89
	1024X1024	23.27	x	x	15	5.53	9	3.97	5	4.11
	2048X2048	143.19	x	x	15	25.49	13	21.52	7	20.96

Robustness of Iterative Solvers

Q: What does it mean?

- Robustness of a system can be viewed as the property of being strong and healthy in constitution
- For solution methods of linear algebraic systems, robustness has two meanings:
  - Breakdown-free and reliable for providing a solution; eg: Robust ILU
  - Performance is resistant to perturbations of parameters

Problem Setting:

$$\max_{P_\alpha \in \mathcal{P}} \left\| \mathrm{Solve}(P_\alpha) \right\| \lesssim \varepsilon$$

- Solve is a solution algorithm (or a set of solution algorithms)
- $\mathcal{P}$ is a given set of problems (preferably parametrized)
- $\| \cdot \|$ is a reasonable performance measurement (0: best, ∞: fail)
- $\| \cdot \|$ should take available resources into account
- ε is a tolerance for worst performance compared with baseline
- $\lesssim$ refers to there might be a constant independent of parameter α

Improving Robustness of Iterative Solvers

Why and where it matters?

- In many applications, we need to solve thousands or more of linear systems from a fixed PDE with a set of physical/discretization parameters
- Physical parameters are nonlinear, heterogenous, anisotropic, degenerating
- Different discretization methods lead to systems with different properties

☞ Classical iterative methods (Weighted Jacobi, SOR, ...) introduce parameters

☞ Solver parameters affect the performance in large extent $\Longrightarrow$ Not robust!

How to improve robustness? Some strategies:

- Combine appropriate iteration, precondition, and decoupling methods
- Improve theoretical understanding for simple model problems and construct preconditioners that are not sensitive to given parameters

☞ Provide an automatic or adaptive procedure to select solver or its parameters to assist simulation software

Combination of Solution Methods

Combining different methods in a single preconditioner

- A framework combines an A-convergent method with an SPD preconditioner
- Porous media flow equation [Hu et al. 2013]
- Radiation diffusion equation [Yue, Shu, Xu, Zhou 2015]
- Another way to combine Schwarz methods [de Dios, Barker, Vassilevski 2014]

Appling multiple solvers at the same time

- Apply Krylov methods with similar structure and combine communications
- Poly-iterative technique [Rice 1967; Barrett, Berry, Dongarra, Eijkhout, Romine 1996]

Composition of different solvers

- A sequence of linear solvers are queued to improve reliability (rate of success)
- Robust composite linear solver [Bhowmick, Raghavan, McInnes, Norris 2004]

Choose a solver from a set of methods automatically $\star$

- Adaptive ILU for CFD simulation [McInnes, Norris, Bhowmick, Raghavan 2003]
- Adaptive AMG setup [Xu, Mo, An 2016; Xu et al. 2020]

Combination of AMG and ILU

An example from ExxonMobil

Preconditioner	#iter	Setup time	Solve time
ILU(0)	3458	0.16	96.19
AMG	362	0.85	40.32
AMG with ILU(0)/1	2255	1.00	305.17
AMG with ILU(0)/2	—	1.18	—
AMG + ILU	47	1.83	12.75

- Robustness of ILU smoother for anisotropic problem [Kettler 1982; Wittum 1989; Stevenson 1994]
- But neither ILU nor AMG works well alone for this test problem
- Efficient ILU smoother for 3D anisotropic problems is difficult to construct?

Question: Why AMG with ILU smoother does not converge?

Consider one V-cycle AMG with an ILU smoother (B denotes ILU and S is CGC)

$$u \leftarrow u + B(f - Au), \quad u \leftarrow u + S(f - Au), \quad u \leftarrow u + B(f - Au).$$

This gives: $I - \bar{B}A = (I - BA)(I - SA)(I - BA)$.

But, in general, $\bar{B}$ might not be positive definite! When does it work?

Effectiveness of Combined Preconditioner

Change the order in which S and B are applied

$$u \leftarrow u + S(f - Au), \quad u \leftarrow u + B(f - Au), \quad u \leftarrow u + S(f - Au).$$

This gives a combined preconditioner $\widetilde{B}$:

$$I - \widetilde{B}A = (I - SA)(I - BA)(I - SA).$$

Theorem (Hu et al. 2013)

Assume that $S : V \to V$ satisfies $\|(I - SA)x\|_A \le \|x\|_A, \forall x \in V$ and that operator $B : V \to V$ is SPD. Then, the operator $\widetilde{B}$ is SPD.

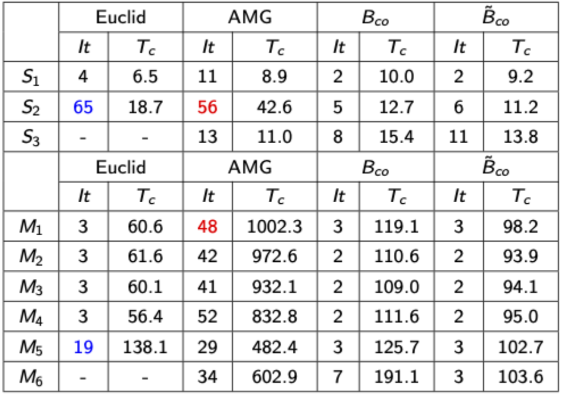
Theorem (Hu et al. 2013)

Assume that $\|(I - SA)v\|_A^2 \le \rho\|v\|_A^2,\ \rho \in [0, 1)$. If B is a SPD and it satisfies that $\lambda_{\max}(BA) > 1 \ge \lambda_{\min}(BA) > 0$, then $\kappa(\widetilde{B}A) \le \kappa(BA)$. Furthermore, if $\rho \ge 1 - \frac{\lambda_{\min}(BA)}{\lambda_{\max}(BA)-1}$, then $\kappa(\widetilde{B}A) \le \kappa(\widetilde{S}A)$ with $\widetilde{S} = S + S^T - S^T AS$.

Remark: $\rho \ge 1 - \frac{\lambda_{\min}(BA)}{\lambda_{\max}(BA)-1}$ means $\rho \approx 1 - \kappa(BA)^{-1}$, if $\lambda_{\max}(BA) \gg 1$.

Nonsymmetric Combined Preconditioner

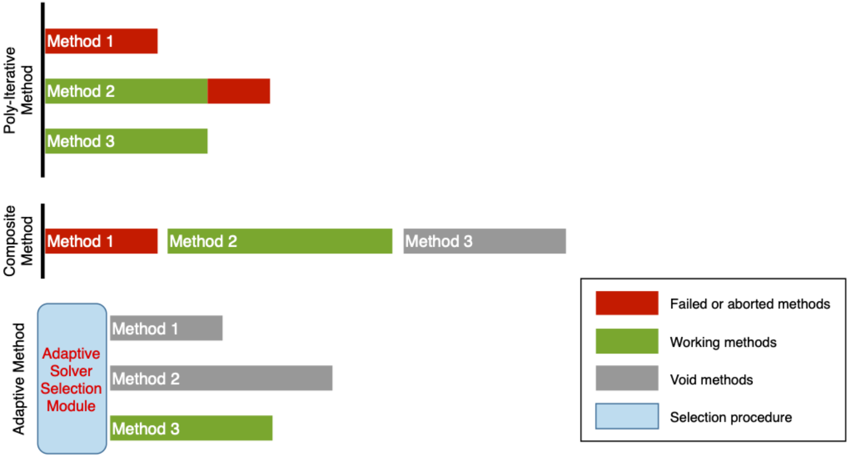
- Approximate $w_1 = A^{-1}g$ by ILU and form the residual $r_1 = g - Aw_1$
- Solve $Aw_2 = r_1$ approximately by one V-cycle and zero as the initial guess
- Compute $\widetilde{B}_{co}g = w_1 + w_2$ [Yue, Shu, Xu, Zhou 2015]



	Euclid		AMG		B_{co}		$\tilde{B}_{co}$	
	It	T_c	It	T_c	It	T_c	It	T_c
S_1	4	6.5	11	8.9	2	10.0	2	9.2
S_2	65	18.7	56	42.6	5	12.7	6	11.2
S_3	-	-	13	11.0	8	15.4	11	13.8
	Euclid		AMG		B_{co}		$\tilde{B}_{co}$	
	It	T_c	It	T_c	It	T_c	It	T_c
M_1	3	60.6	48	1002.3	3	119.1	3	98.2
M_2	3	61.6	42	972.6	2	110.6	2	93.9
M_3	3	60.1	41	932.1	2	109.0	2	94.1
M_4	3	56.4	52	832.8	2	111.6	2	95.0
M_5	19	138.1	29	482.4	3	125.7	3	102.7
M_6	-	-	34	602.9	7	191.1	3	103.6

Number of iterations and wall time (sec) of right-preconditioned GMRES(30) solvers

Comparison of Combined Solution Methods



1. Poly-iterative methods: apply multiple solvers simultaneously
2. Composite methods: apply multiple solvers sequentially (dynamic ordering)
3. Automatic solver selectors: pick a solver based on some criteria

Choosing Solver Parameters

For any problem $P_\alpha \in \mathcal{P}$, find a solver $S_\beta \in \mathcal{S}$ such that $\|S_\beta(P_\alpha)\| \lesssim \varepsilon$

Available Solvers

Black-box solvers

Efficiency
Scalability

Gray-box solvers

White-box solvers

Robustness
Applicability

Problems

- Type of PDE (system)
- Physical parameters
- Discretization
- ......

Algorithms

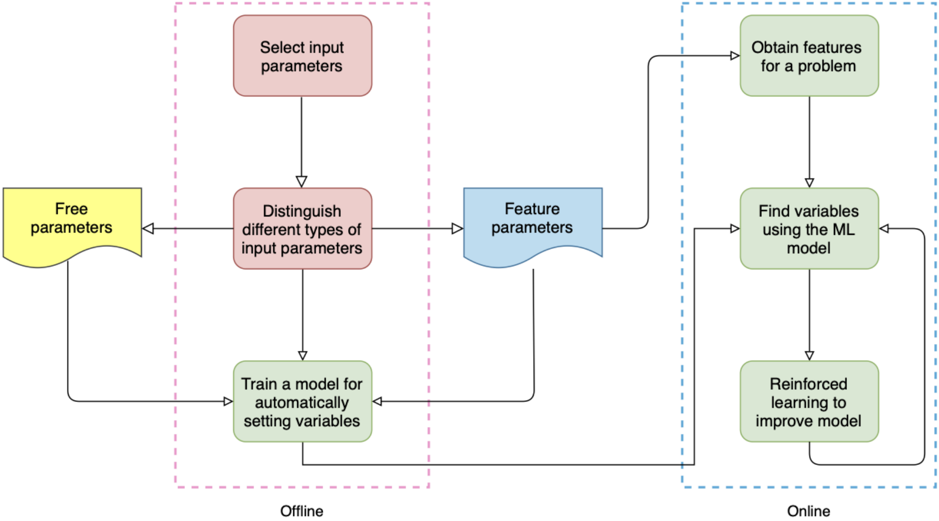
- Krylov methods
- Preconditioning
- Decoupling
- ......

Resources

- Num of nodes
- Num of processes
- Num of threads
- ......

$$\text{Parameters} \xrightarrow{\text{analysis}} \text{Feature (fixed)} \cup \text{Free (variable)} \xrightarrow{\text{tuning}} \text{Performance!}$$

Workflow to Construct Adaptive Solvers


Select input parameters
Distinguish different types of input parameters
Free parameters
Feature parameters
Train a model for automatically setting variables
Obtain features for a problem
Find variables using the ML model
Reinforced learning to improve model
Offline
Online

Need to begin with a general enough framework (KSM + PC, e.g. AMG, DDM, ILU)!

Automatic Solver Selector

Find the optimal solver from the available ones: $\operatorname{argmin}_{S_\beta \in \mathcal{S}} \max_{P_\alpha \in \mathcal{P}} \|S_\beta(P_\alpha)\| \lesssim \varepsilon$

- Automatic and adaptive procedures for select solver or solver parameters
  - Automatic procedure: give a mapping from $\mathcal{P}$ to $\mathcal{S}$
  - Adaptive (or self-adaptive) procedure: this mapping also evolves
- Timing for automatic tuning in general

Offline: Computational pattern is independent of the user data

Online: Optimal choice depends largely on the user data, run-time tuning

Hybrid: Combines offline and online tuning steps

Compile ⟶ Compute ⟶ Collect profile ⟶ Compile again

- Key components:
  - A class of algorithms available
  - ☞ A performance model (input, output, practical, accurate)
  - An automated analyzer of problem
  - A self-adaptation strategy

Choosing Feature Parameters

Feature names	
avgnnzprow	right-bandwidth
avgdistfromdiag	symmetry
n-dummy-rows	blocksize
max-nnzeros-per-row	diag-definite
lambda-max-by-magnitude-im	lambda-max-by-magnitude-re
ellipse-cy	nnzup
ruhe75-bound	avg-diag-dist
nnz	left-bandwidth
lambda-min-by-magnitude-im	lambda-min-by-magnitude-re
norm1	sigma-min
upband	n-struct-unsymm
colours	diagonal-average
diagonal-dominance	dummy-rows
ritz-values-r	symmetry-snorm
symmetry-fanorm	symmetry-fsnorm
lambda-max-by-real-part-im	lambda-max-by-real-part-re
lambda-max-by-im-part-re	lambda-max-by-im-part-im
col-variability	trace-abs
ritz-values-c	nnzeros
diag-zerostart	loband
positive-fraction	trace
min-nnzeros-per-row	diagonal-sign
row-variability	nrows
colour-offsets	n-colours
relsymm	diagonal-variance
departure	nnzlow
n-nonzero-diags	sigma-max
dummy-rows-kind	kappa
n-ritz-values	colour-set-sizes
sigma-diag-dist	symmetry-anorm
ellipse-ax	ellipse-ay
ellipse-cx	lee95-bound
normInf	normF
nnzdia	trace-asquared

Full feature set comprising of 68 features computed using Anamod

Feature learning (selection & extraction)

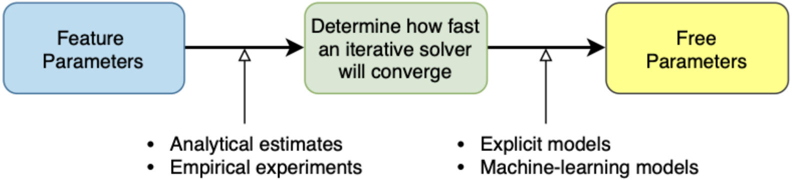
- ☞ Performance of ML model is largely affected by the choice of features
- ● A comprehensive feature set includes all entries of A^{-1} (too complicated)
- ● The choice of features is usually problem-dependent
- ☞ Obtaining features is usually costly and needed in both offline and online steps
- ● Training takes a lot of time if too many features are selected
- ● Analytical and empirical results should be used to select features

Key: Choosing a good set of features is essential!

Performance Model for Solver Selection

- Choose a general enough algorithm framework which is efficient or even optimal for simple cases and can be adjusted for more difficult cases
- Find a small set of feature parameters which affect solver performance the most
- Construct a performance model based on analytical convergence factor estimates or empirical experiments which can predict how efficient the solver might be

☞ Train a performance model efficiently based on actual simulation runs, in case an analytical performance model is hard or not possible to be obtained



Key: Finding a small but influential set of (feature and free) parameters!

Classical Models for Solver Selection

1. Based on physical parameters (diffusion coefficient, Peclet number, ...)
2. Based on PDE type (elliptic, transport, ...)
3. Based on discretization method (SUPG, EAFE, ...)
4. Based on convergence stage in nonlinear iteration (early, near convergence, ...)
5. Based on phase during transient process (based on physics ...)
6. Based on structural features (problem size, symmetry, positivity, ...)
7. Based on norm-related features (trace, L^1-norm, ...)
8. Based on spectral features (condition number, numerical range, ...)
9. Based on variability features (row variability, diagonal dominance, ...)

Key: Finding an adaptive procedure to automatically pick free parameters!

Software for Selecting Algorithms Automatically

ESI group

- The Equation Solver Interface: develop an integral set of standards for solver components
- Multi-lab working group & interface design effort hosted by Sandia, 1997

LSA project

- Linear System Analyzer: build a problem solving environment for linear systems
- http://www.extreme.indiana.edu/pseware/lsa/index.html, Bramley, Gannon, et al. 1998

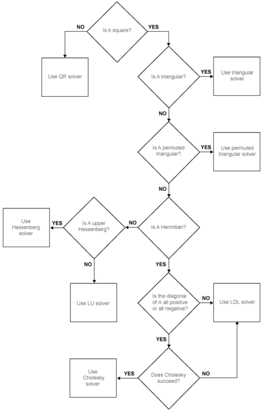
SANS project: SALSA, AnaMod

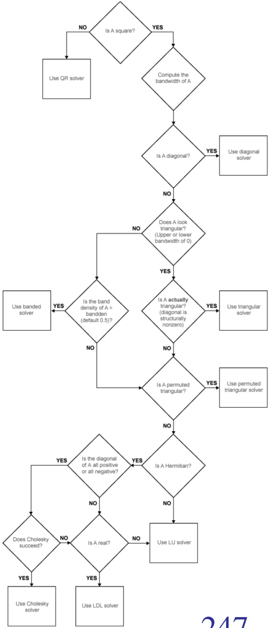
- SANS: Self-Adapting Numerical Software
- SALSA: Self-Adapting Large-scale Solver Architecture
- http://icl.cs.utk.edu/salsa/, Demmel, Dongarra, Eijkhout, et al. 2002

ASLib project

- Algorithm Selection Library: A benchmark library for algorithm selection in general
- Algorithm selection problem [Rice 1976]
- http://www.coseal.net/aslib/, Bischl, Kerschke, et al. 2016

Taxonomy Solver Selection: An Example




Is A square?
NO
YES
Use QR solver
Compute the bandwidth of A
Is A diagonal?
YES
Use diagonal solver
NO
Does A look triangular? (Upper or lower bandwidth of 0)
NO
YES
Is the band density of A > bandden (default 0.5)?
YES
Use banded solver
NO
Is A actually triangular? (diagonal is structurally nonzero)
YES
Use triangular solver
NO
Is A permuted triangular?
YES
Use permuted triangular solver
NO
Is A Hermitian?
YES
NO
Is the diagonal of A all positive or all negative?
YES
NO
Does Cholesky succeed?
NO
YES
Use Cholesky solver
Is A real?
NO
YES
Use LU solver
Use LDL solver

- mldivide: Automatic taxonomy solver in Matlab
- Solver algorithms:

  QR
  LU
  LDLt
  Cholesky
  Banded
  Triangular
  Diagonal
  ⋮

- Robust and easy to use

Taxonomy Solver Selection: Another Example

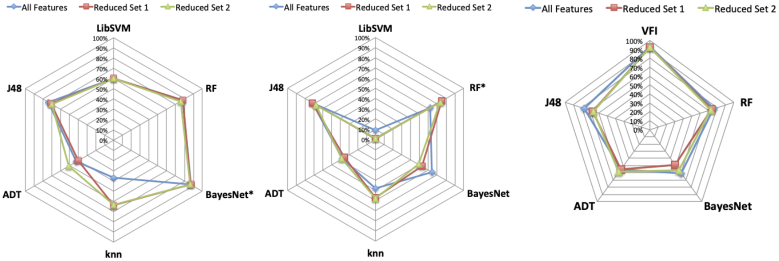
Existing tools http://lighthousehpc.github.io/lighthouse/

- LAPACK search engine: dense linear algebra
- Lighthouse: iterative solvers for LAPACK / PETSc / SLEPc / Trilinos

Functionalities: Create performance model using classification algorithms

- Input: problem characteristics or coefficient matrix
- Output: calling prototype or a piece of code

☞ Disadvantages: Features too algebraic (sym, norm1, nnz, min-nnz-per-row, ...)



Classification Accuracy. Left: PETSc+AnaMod; Middle: Trilinos+AnaMod; Right: Trilinos.

Three Steps Towards An Adaptive Solver

I. From experience to automated procedure

> Using theoretical or practical info to set range of algorithms/parameters

- Reduce number of input (feature and free) parameters — Key!
- Many machine learning methods can be used to get an initial model

II. Performance enhancement during application

> Improving the initial model by reinforced learning during practice

- Every application has its own characteristics
- The initial training set might not have enough data from this application

III. General-purpose iterative solver package

> Make a general-purpose iterative solver package with build-in parameter autotuning and self-learning mechanism

- Include state-of-the-art algorithms enhanced with a robust autotuner
- Like a black-box iterative solver with expanding abilities

Step I. From Experience to Automated Procedure

For a given task, we have the following workflow:

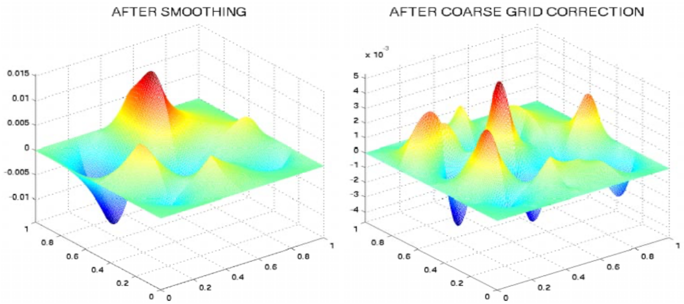
(i) find a suitable algorithm framework which has good properties;

(ii) choose input parameters;

(iii) gather training/validation/test data;

(iv) train a good initial machine learning model;

(v) improve the model during application.

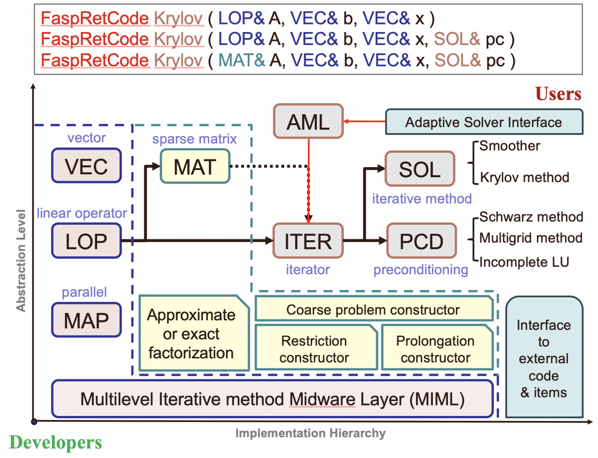


Step Ⅱ. Performance Enhancement during Application

After obtaining an initial model, find suitable transfer learning and reinforced learning techniques for a new task and improve the model during application.

Step Ⅲ. General-Purpose Iterative Solver Package

Make a package which not only contains enough algorithm building blocks, but also predicts good parameters according to problem characteristics, and most importantly evolve ...



```
FaspRetCode Krylov ( LOP& A, VEC& b, VEC& x )
FaspRetCode Krylov ( LOP& A, VEC& b, VEC& x, SOL& pc )
FaspRetCode Krylov ( MAT& A, VEC& b, VEC& x, SOL& pc )
```