Multilevel Iterative Methods And Applications

Chensong Zhang

NCMIS & LSEC, Chinese Academy of Sciences

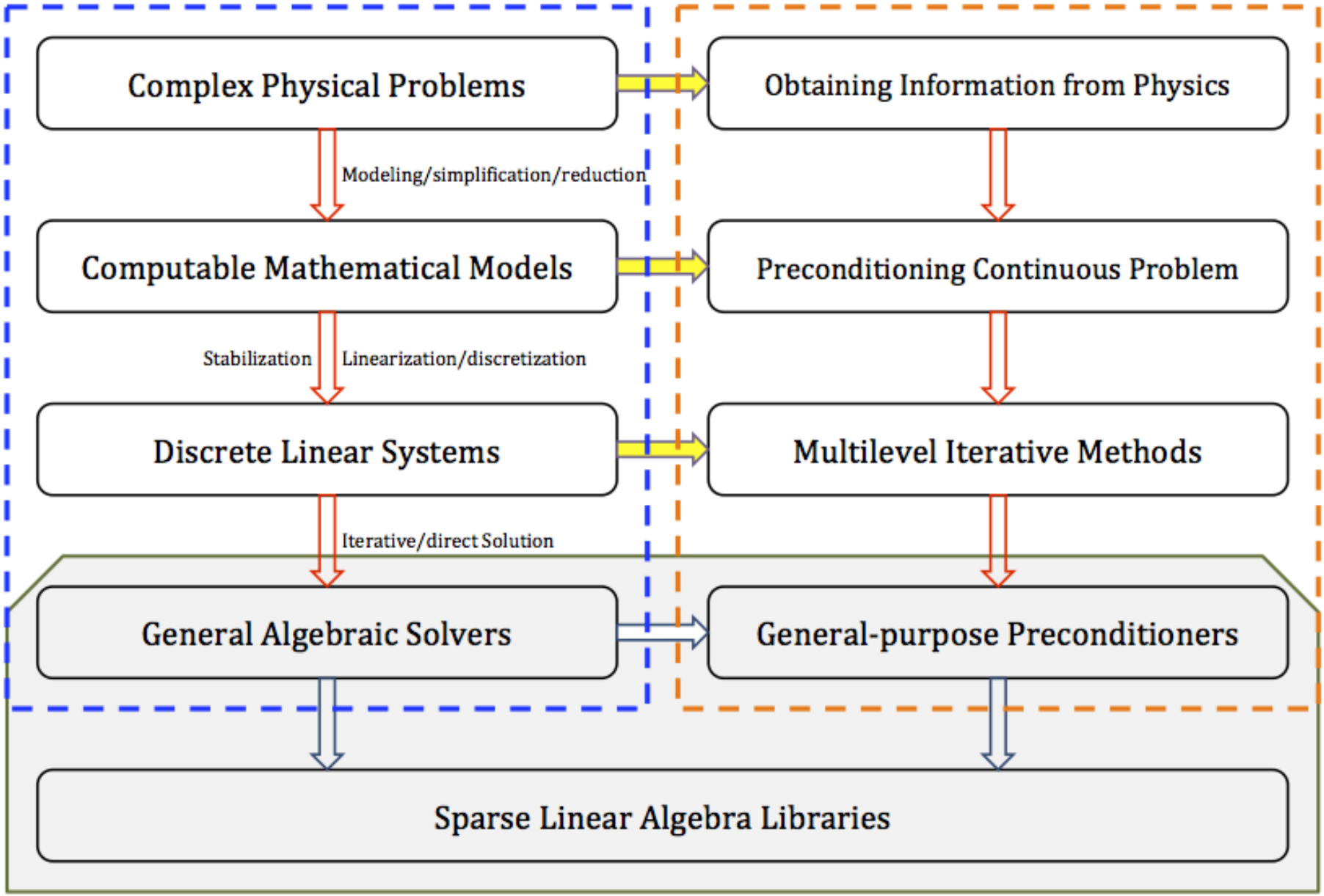
LSEC, Beijing — Oct 21/28, 2016

Outline

1. Introduction
2. Multilevel Iterative Methods
3. Applications in Fluid Problems
4. Applications in Fluid-Structure Interactions
5. Applications in Petroleum Reservoir Simulation
6. Applications in Optimization Problems
7. Error-Resilient Multilevel Iterative Methods

1. Introduction

Numerical Simulation of Physical Problems



Linear Algebraic Solvers

A fundamental problem in scientific computing:

> Given a sparse matrix $A \in \mathbb{R}^{N \times N}$ and $b \in \mathbb{R}^N$, solve $Ax = b$!

In many applications, it takes most of the simulation time!

- General purpose direct solvers: Gaussian Elimination, ...
  - Robust, exact, multiple right-hand sides, ...
  - Black Box $\Longrightarrow$ Many packages available: PARDISO, MUMPS, SPOOLES, SuiteSparse (CHOLMOD/UMFPACK), SuperLU, WSMP, H2Lib, ...
  - Memory: Require explicit matrices, need more RAM for decomposition
  - Computation: General $\mathcal{O}(N^3)$, banded $\mathcal{O}(N^2)$, nested dissection $\mathcal{O}(N^{1.5})$ [George 1973; Duff-Erisman-Reid 1986; Demmel 1997]
- $\mathcal{H}$-matrix, data-sparsity, low-rank approximation: [Hackbush 1999; Chandrasekharan-Gu-Lyons 2005; Xia-Chandrasekharan-Gu-Li 2009, 2010; Ho-Greengard 2012; Schmitz-Ying 2012; ...]
- Specialized methods: FFT, ...

Iterative Solution Methods

Pros:

- Optimal cost is possibly: $\mathcal{O}(N|\log N|^{\sigma})$ operations
- Adjustable accuracy with good initial guess in practice
- Matrix-free operations can be used
- Singular or nearly singular problems

Cons:

- Problem-dependence: require different methods for different problems
- Robustness: (arguably) biggest disadvantage in practice
- Optimality: optimal algorithm or fastest algorithm?
- Implementation: difficult if not impossible to make efficient software

Goals:

> convergence, robustness, optimality, efficiency, scalability, reliability

Whatever Has Been Done Can Be Outdone

Obstacles to scalable simulation

- Model complexity
- Extreme concurrency
- Multiple levels of parallelism
- Complex memory hierarchies
- More costly data movement
- Hardware failures and soft error

- Optimization: Improve cooling, find hot spots, reduce power leakage
- Less transistors $\Longrightarrow$ lower frequency $\Longrightarrow$ more processing cores
- Scalable, power-aware, resilient parallel algorithms and software

Krylov Subspace Methods and Preconditioners

1. Conjugate gradient method for $Au = f$

$$\frac{\|u - u^m\|_A}{\|u - u^0\|_A} \leq 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^m \ (m \geq 1), \quad \kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

2. Convergence estimate using effective condition number

$$\frac{\|u - u^m\|_A}{\|u - u^0\|_A} \leq 2C\left(\frac{\sqrt{b/a} - 1}{\sqrt{b/a} + 1}\right)^{m-m_0} \ (m \geq m_0)$$

   - decomposition: $\sigma(A) = \sigma_0(A) \cup \sigma_{\text{eff}}(A)$ with m_0 entries in $\sigma_0(A)$
   - effective condition number $\kappa_{\text{eff}}(A) := b/a = \lambda_N(A)/\lambda_{m_0+1}(A)$
   - constant $C := \max_{\lambda \in \sigma_{\text{eff}}(A)} \prod_{\mu \in \sigma_0(A)} \left|1 - \frac{\lambda}{\mu}\right| \leq \left(\kappa(A) - 1\right)^{m_0}$

☞ Preconditioners:
   - Incomplete factorizations: ILUk, ILUt, ILUtp, ...
   - Domain decomposition methods: RAS, FETI, BDDC, ...
   - Multilevel preconditioners: AMG, GAMG, GMG, ...

Designing Multilevel Solver Software

1. Existing packages with MG methods: hypre, ML, AGMG, ...
2. How to handle a complicated PDE (system)?
3. How to choose a discretization?
4. How to handle discretizations on unstructured mesh?

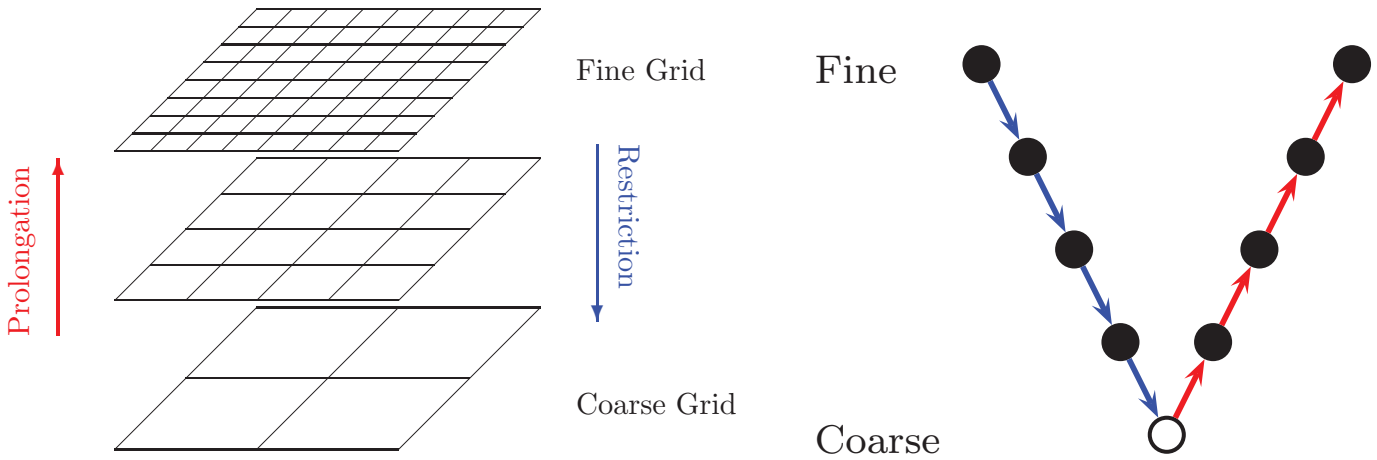
2. Multilevel Iterative Methods

Multilevel Iterative Methods

Examples of multilevel algorithms

- Quick Sort, FFT, FMM, GMG, AMG, H-Matrix, H^2-Matrix, ...

Multigrid V-cycle



Key gradients for multilevel iterative methods

- ✏ Construct multilevel hierarchy in an efficient way
- ✏ Find effective (and cheap) smoothers for each level
- ✏ Find good coarser level solvers (nested iterations)

An Example: Multigrid Method

Performance comparison: GMG vs AMG

Solution Method	FMG	GMG-PCG	CA-PCG	UA-PCG
Number of Iterations	—	5	6	12
Wall Time (sec)	0.143	0.251	1.57 (0.87)	1.50 (0.26)

Table: Solving 2D Poisson's equation using multigrid methods (Five-point stencil, FASP 1.8.3, DOF = 1M, TOL = 10^{-6}, Macbook Pro 13', gcc-4.9.3, -O2)

Methods based on PDE and/or discretization information

- Using connectivity information from coefficient matrix (AMG)
- Using an extended matrix (Jacobi = BPX, GS = MG V-cycle)
- Using an auxiliary grid or discretization
- Using coarsening based on the finest grid
- Block preconditioners for coupled PDEs

Must plan ahead of time: meshing, linearization, discretization, ...

Method of Subspace Corrections

Divide and conquer

- Space decomposition: $V = \sum_{i=1}^{n} V_i$
- Subspace correction: $e_i \approx A_i^{-1} P_i (f - Au)$

$$u \leftarrow u + \sum_{i=1}^{n} e_i \qquad \text{(Parallel subspace corrections, Jacobi)}$$

$$u \leftarrow u + e_i, \ i = 1 : n \qquad \text{(Successive subspace corrections, GS)}$$

Some examples and generalizations

- BPX preconditioner [Bramble-Pasciak-Xu 1990]
- SIAM Review [Xu 1992]
- Fictitious domain method [Nepomnyaschikh 1992]
- Auxiliary space method [Xu 1996]
- Nonlinear equations [Tai-Xu 2002]
- $H(\mathrm{div})$, $H(\mathrm{curl})$ solvers [Hiptmair-Xu 2007]

Designing Multilevel Solver Software

1. How to handle a complicated PDE (system)?
   - Provide blockwise iterative methods and general preconditioners
   - Use mapping properties to construct a good preconditioner
   - Reduction: precond PDE systems $\Longrightarrow$ precond model problems
   - Use an auxiliary problem for preconditioning
2. How to choose a discretization?
   - Using a uniform stable discretization is important
   - Using a solver-friendly discretization
3. How to handle discretizations on unstructured mesh?
   - To improve efficiency of the SETUP phase of multilevel methods
   - Use sparsity pattern or entries of coefficient matrices
   - Use an auxiliary structured (or semi-structured) grid

Preconditioned Krylov Methods in Hilbert Space

1. What about more general problems with $A : X \to X' \supset X$?
   - Need to construct a SPD $B : X' \to X$ to make KSM's to work
   - If A is SPD, then $\langle \cdot, \cdot \rangle_A := \langle A\cdot, \cdot \rangle$ defines an inner-product and
     $$\langle BAx, y \rangle_A = \langle ABAx, y \rangle = \langle Ay, BAx \rangle = \langle Ax, BAy \rangle = \langle x, BAy \rangle_A$$
   - BA is a SPD in terms of $\langle A\cdot, \cdot \rangle$ or $\langle B^{-1}\cdot, \cdot \rangle$
   - Convergence estimate of CG holds true with $\kappa(BA)$
2. Find a natural (canonical) preconditioner for continuous problem
   - Bilinear form $a : X \times X \to \mathbb{R}$ is symmetric and bounded, and it satisfies the inf-sup condition $\inf_{x \in X} \sup_{y \in X} \frac{a(x,y)}{\|x\|_X \|y\|_X} \geq \gamma > 0$
   - For $f \in X'$, let $B : X' \to X$ be a Riesz operator $(Bf, y)_X = \langle f, y \rangle$
   - Then $BA : X \to X$ is symmetric in $(\cdot, \cdot)_X$ and $\kappa(BA) \leq C_a/\gamma$!
     - $\|BA\| \leq \sup_{x \in X} \frac{(BAx,x)_X}{\|x\|_X^2} = \sup_{x \in X} \frac{|a(x,x)|}{\|x\|_X^2} \leq C_a$
     - $\|(BA)^{-1}\|^{-1} = \inf_{x \in X} \frac{\|BAx\|_X}{\|x\|_X} = \inf_{x \in X} \sup_{y \in X} \frac{a(x,y)}{\|x\|_X \|y\|_X} \geq \gamma$

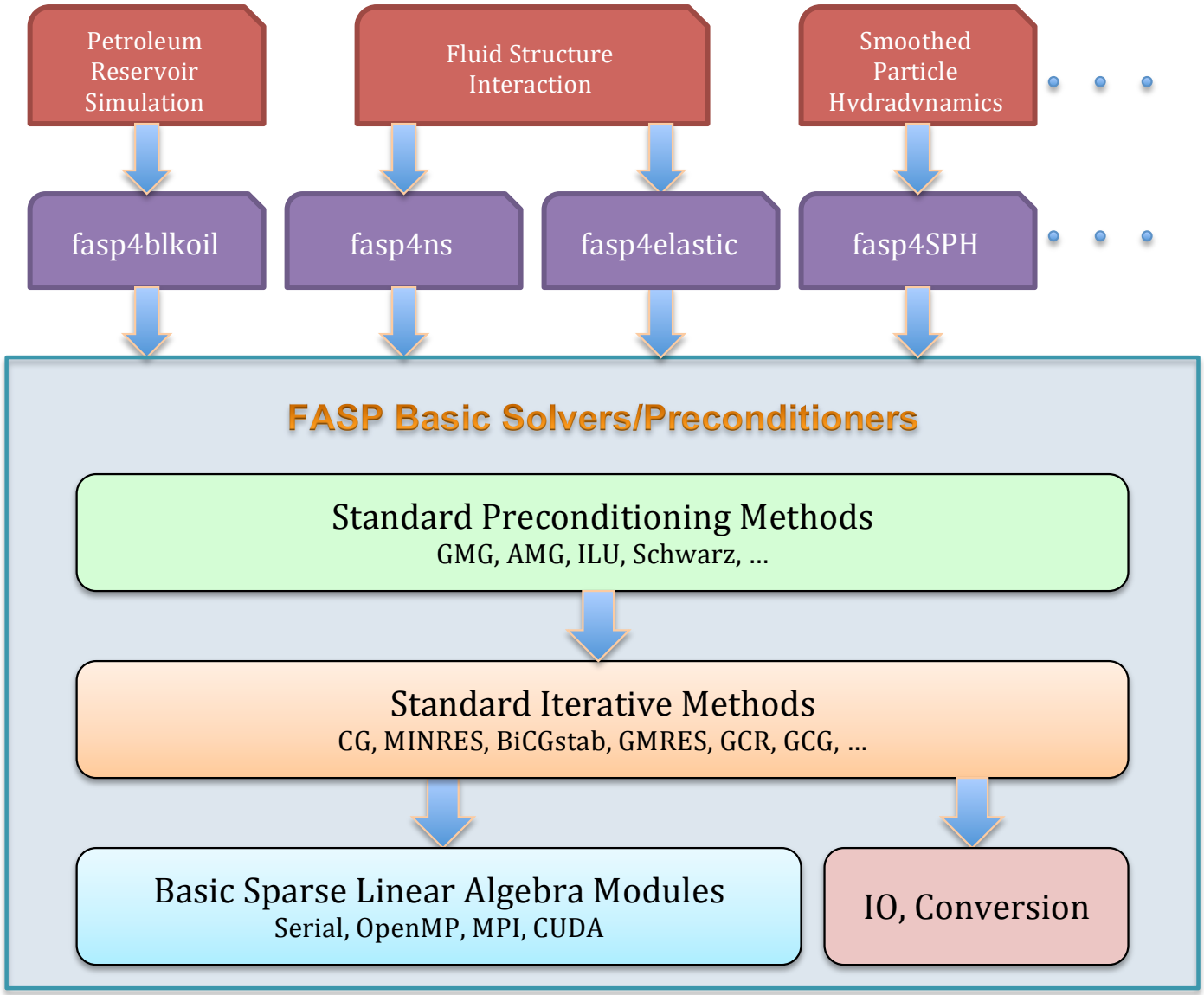
[Mardal-Winther 2011]

Construction of Preconditioners

1. What does a "natural" preconditioner look like?
   - Let $A : H_0^1(\Omega) \to H^{-1}(\Omega)$ s.t. $\langle Au, v\rangle := a(u,v) = \int_\Omega (\alpha(x)\nabla u)\cdot\nabla v\, dx$
   - Kernel $\alpha(x) \in \mathbb{R}^{d\times d}$ satisfies $\gamma|\xi|^2 \le \xi^T\alpha(x)\,\xi \le C_a|\xi|^2$
   - Define $B = (-\Delta)^{-1} : H^{-1}(\Omega) \to H_0^1(\Omega) \implies \kappa(BA) \le C_a/\gamma < \infty$
   - Stokes problem: $a\big((u,p),(v,q)\big) := \langle\nabla u, \nabla v\rangle + \langle p, \nabla\cdot v\rangle + \langle q, \nabla\cdot u\rangle$
   - $A : \big[H_0^1(\Omega)\big]^d \times L_0^2(\Omega) \to \big[H^{-1}(\Omega)\big]^d \times L_0^2(\Omega)$

     $\implies B = \mathrm{diag}\big[(-\Delta)^{-1}, \ldots, (-\Delta)^{-1},\ I\big] \implies$ Block Trig Precond, ...
2. Solve a discrete problem $\implies$ Employ a stable discretization
   - Stable discretization, i.e., $\inf_{x\in X_h}\sup_{y\in X_h}\frac{a(x,y)}{\|x\|_X\|y\|_X} \ge \gamma_1 > 0$
   - Condition number can be bounded $\kappa(B_hA_h) \le C_a/\gamma_1$
3. Construct a cheap spectral-equivalent preconditioner
   - Discretization, grid generation/adaptation, parallelization, ...
   - Example: need components like $(-\Delta)^{-1}$ when solving Stokes, Darcy, ...

FASP Software Project



Preliminary Tests: AMG (Sequential)

Test Device: Intel Core i5 2.6GHz, 8GB RAM, gcc 4.9.2 -O2
Benchmark: FASP 1.7.0, hypre 2.10.0b, AGMG 3.2.0 (default parameters)

Problem	DOF	RS-V-CG	UA-NA-CG	hypre	AGMG
2D 5pt	1M	1.79	1.43	1.96	1.73
2D 5pt	4M	8.71	6.04	8.45	6.61
2D 9pt	1M	1.82	2.07	2.25	2.24
2D 9pt	4M	7.63	8.39	8.88	9.42
3D 7pt	$\frac{1}{4}$M	1.05	0.37	1.83	0.43
3D 7pt	2M	10.86	3.28	19.04	3.71
3D 27pt	$\frac{1}{4}$M	2.09	0.94	3.26	1.79
3D 27pt	2M	20.0	8.53	34.54	20.29

Table: Computing time (seconds) of the AMG-preconditioned conjugate gradient method. We solve the 2D/3D Poisson equation with one processing core. Stopping criteria: relative residual is less than 10^{-6}.

Preliminary Tests: AMG (OpenMP)

Test Device: Intel Xeon X5675 3.07GHz (6 cores), 24GB RAM, gcc 4.4.6

–O2	DOF	NT=1	NT=2	NT=4	NT=8	NT=12
2D 5pt	1M	2.12s	×1.43	×1.74	×1.90	×1.89
2D 5pt	4M	9.51s	×1.47	×1.80	×2.00	×2.00
3D 7pt	2M	9.86s	×1.50	×1.89	×2.15	×2.20
3D 7pt	16M	90.66s	×1.50	×1.90	×2.20	×2.29
–O0	DOF	NT=1	NT=2	NT=4	NT=8	NT=12
2D 5pt	1M	6.36s	×1.54	×2.15	×2.60	×2.62
2D 5pt	4M	27.14s	×1.56	×2.22	×2.70	×2.70
3D 7pt	2M	31.00s	×1.59	×2.25	×2.87	×3.07
3D 7pt	16M	274.3s	×1.64	×2.35	×3.01	×3.28

Table: Computing time (seconds) of the classical AMG method. We solve the 2D/3D Poisson equation with OpenMP. Stopping criteria: relative residual is less than 10^{-6}.

Preliminary Tests: GMG (CUDA)

CPU: AMD 2.8GHz 8-core (using a single core), gcc 4.4.6 –O2
GPU: NVIDIA GTX480 480 cores 1.5GB RAM ($485), nvcc 4.1 –O2

Table: Kernel time (seconds) in 2D case

DOF	FFTW	FMG(1,2)	CUFFT	FMG(1,2)
1M	0.260	0.108	0.0110	0.0088
4M	2.020	0.452	0.0408	0.0257
16M	6.650	1.830	0.1364	0.0917

Some observations

- $15\times \sim 18\times$ speed-up compared with single-thread CPU version (2/3D)
- Speedup of GMG is not as good as FFT (almost $50\times$)
- GMG on GPU: 15GFlops, only 10% of peak performance
- Bottleneck: Visiting coarse level spaces
- Solution: BPX + Redundant Basis Formulation

[Feng-Shu-Xu-Z. 2014]

3. Applications in Fluid Problems

Model Problems

Incompressible Viscid Newtonian Fluids:

$$\begin{cases} \mathrm{u}_t + \mathrm{u}\cdot\nabla\mathrm{u} - \mu\Delta\mathrm{u} + \nabla p = 0 & \Omega & \text{momentum equation} \\ \nabla\cdot\mathrm{u} = 0 & \Omega & \text{incompressibility} \\ \mathrm{u} = 0 & \partial\Omega & \text{no-slip boundary} \end{cases}$$

A model problem:

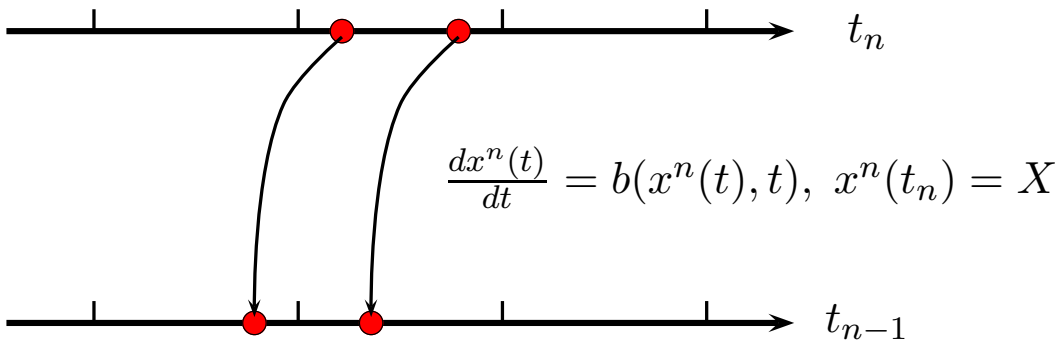
$$u_t + b(x,t)\cdot\nabla u - \mu\Delta u = f \qquad \text{for } t \in (0, T]$$

Numerical methods:

- Simple upwind [Courant-Isaacson-Rees 1952; Greenspan 1968]
- ENO/WENO [Harten et al. 1987; Liu-Osher-Chan 1994]
- Upwind finite element [Christie et al. 1976; Heinrich et al.1977]
- Streamline upwind [Hughes-Brook 1982]
- RKDG [Cockburn-Shu 1991; Chen-Cockburn-Jerome-Shu 1995]
- LDG [Bassi-Rebay 1997; Cockburn-Shu 1998]

Method of Characteristics

- Eulerian-Lagrangian method
  - Semi-Lagrangian [Wiin-Nielsen 1959; Robert 1981, 1982]
  - Characteristic-Galerkin [Douglas-Russell 1982; Pironneau 1982]
  - Convergence and nonlinear stability for NS [Süli 1988]
- $\frac{\partial u}{\partial t} + b(x,t) \cdot \nabla u \approx \frac{U^n - U_*^{n-1}}{k}$



- Advantages
  - Stable and solver-friendly
  - Uniform treatment for the nonlinear term
  - Feet searching is embarrassingly parallelizable

Solvers for the Stokes Equation

1. Iterative methods: MINRES, GMRES, GCR, ...
2. We use block preconditioners motivated by

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} I_u & 0 \\ BA^{-1} & I_p \end{pmatrix} \begin{pmatrix} A & B^T \\ 0 & -S \end{pmatrix}$$

$$= \begin{pmatrix} I_u & 0 \\ BA^{-1} & I_p \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & -S \end{pmatrix} \begin{pmatrix} I_u & A^{-1}B^T \\ 0 & I_p \end{pmatrix}$$

3. About Stokes preconditioners
   - Steady-state: Bramble-Pasciak 1988; Rusten-Winther 1992
   - Time-dependent: Vanka 1986; Bramble-Pasciak 1997; Braess-Sarazin 1997; Shen-Xu 1998; Peters-Olshanskii-Reusken 2006
   - Parallel implementation: May-Moresi 2008; Geenen et al. 2009; ...
   - Survey: Benzi-Golub-Liesen 2005; Larin-Reusken 2009; He-Viuk 2016
4. DD methods: Bramble, Cai, Dohrmann, Girault, Kim, Klawonn, Lee, Li, Pasciak, Pavarino, Riviere, Widlund, Zampini, ...

[Toselli-Widlund 2004]

Robustness of AMG and GAMG

Method (θ)	1 Core			8 Cores			64 Cores		
	#It	CPU	RAM	#It	CPU	RAM	#It	CPU	RAM
AMG(0.2)	19	42.47	1986	28	73.22	2043	93	438.8	2352
GAMG(0.2)	16	16.45	1023	18	20.03	1117	17	25.77	1534
AMG(0.4)	16	22.12	1560	36	57.40	1840	>500	>800	2258
GAMG(0.4)	16	16.41	1023	18	20.05	1117	17	25.10	1535
AMG(0.6)	17	16.44	1679	25	33.83	1683	245	356.2	2085
GAMG(0.6)	16	16.42	1023	18	19.07	1117	17	25.64	1539
AMG(0.8)	17	13.08	1319	19	22.79	1555	19	36.50	1971
GAMG(0.8)	16	16.47	1023	18	20.93	1118	17	25.23	1534

Table: Iteration number, CPU time, and memory usage of the AMG and GAMG preconditioned CG method for the 3D Poisson equation on unstructured tetrahedral mesh ($P^{4,0}$ finite element, about 500K DOF per processing core).

1. One core and many cores behave quite differently
2. AMG becomes less robust for higher order FEM
3. Less memory per core $\Longrightarrow$ save memory and less data movement

Block Preconditioner for the Stokes Equation

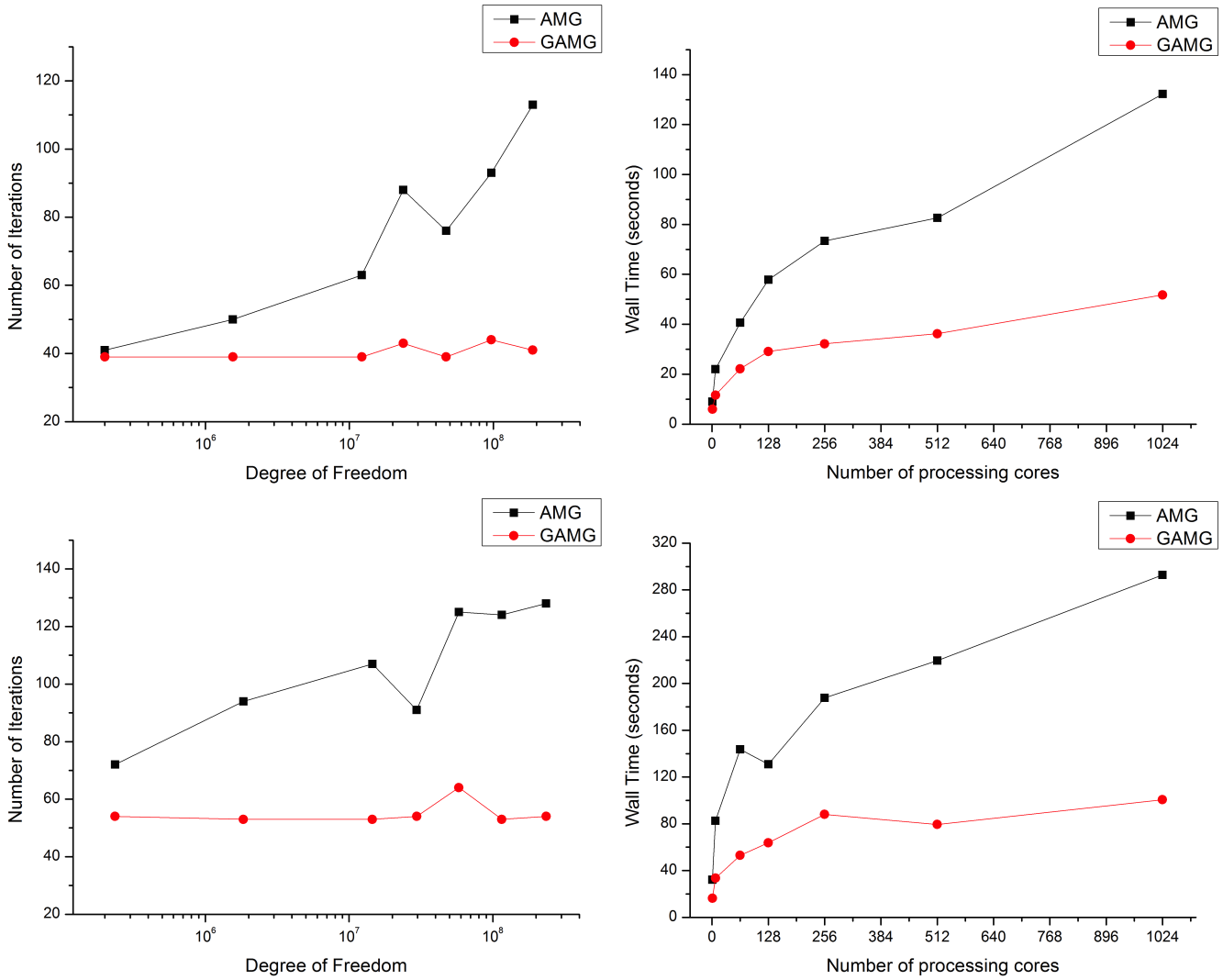


Figure: The upper two: $P^{3,0}$–$P^{2,0}$ and the lower two: $P^{4,0}$–$P^{3,0}$

Oldroyd-B Model

A Simplified Non-Newtonian Model

$$
\begin{aligned}
-\mu_s \Delta \mathrm{u} + \nabla p &= \nabla \cdot c \\
\nabla \cdot \mathrm{u} &= 0 \\
c + \mathrm{Wi}\left[\frac{\partial c}{\partial t} + \mathrm{u} \cdot \nabla c - \nabla \mathrm{u}\, c - c\,(\nabla \mathrm{u})^T\right] &= \frac{\mu_p}{\mathrm{Wi}} I
\end{aligned}
$$

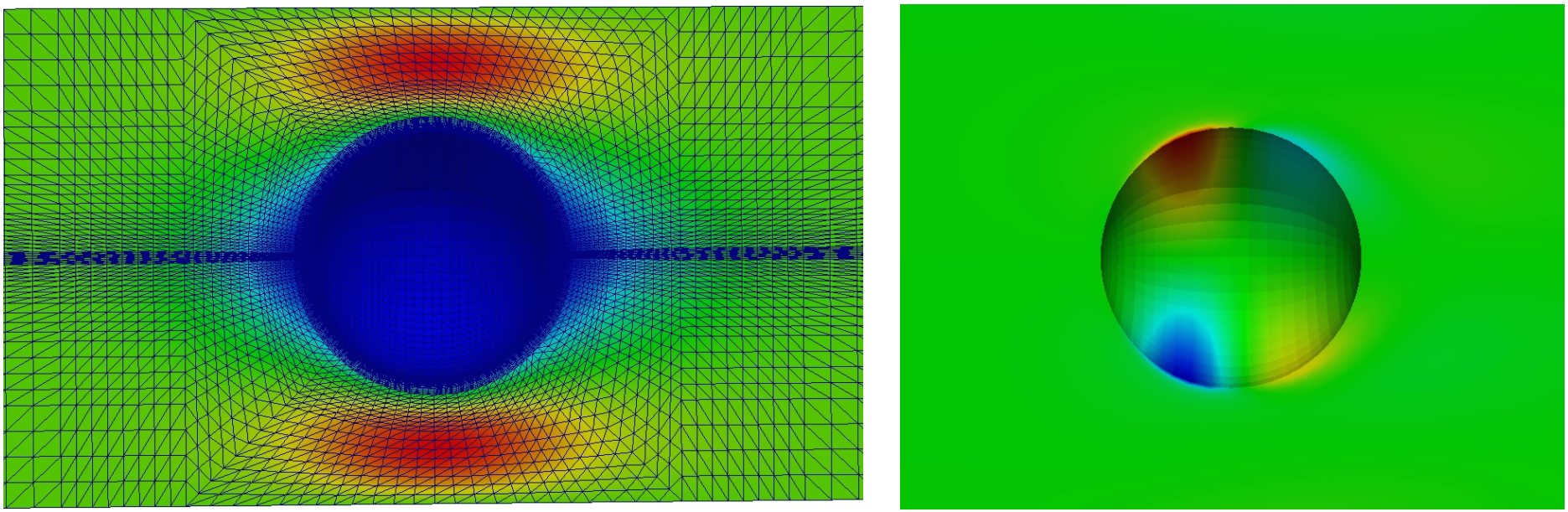


Figure: 3D Oldroyd-B with Wi $= 0.6$: velocity field (left) and conformation tensor c_{yz} (right)

Weak Scalability Test

Table: Scalability test for 3D Oldroyd-B benchmark problem

#Procs	#Elems	DOF(u, p)	#It GMRES	Stokes (sec)	DOF(c)	MoC (sec)
8	1,074,143	4,574,383	60	40	104,206,176	107
16	2,942,264	12,618,793	61	74	284,966,802	108
32	5,085,792	22,209,594	64	74	494,746,218	105
64	8,593,144	36,197,957	54	69	829,285,614	88
128	19,894,022	84,835,667	55	119	1,922,481,522	102
256	39,351,715	172,014,185	50	121	3,824,031,384	106
512	68,745,152	288,007,002	49	134	6,616,889,802	111

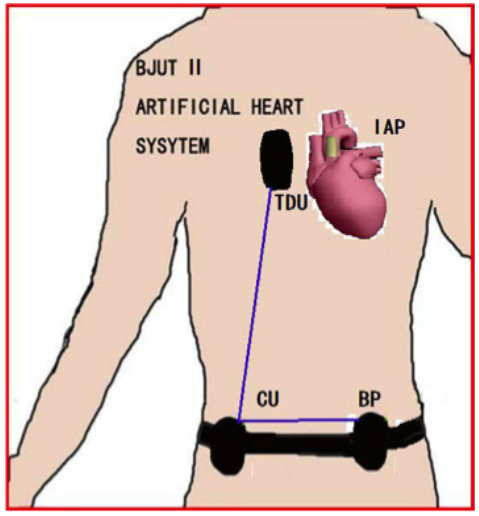
Numerical Method

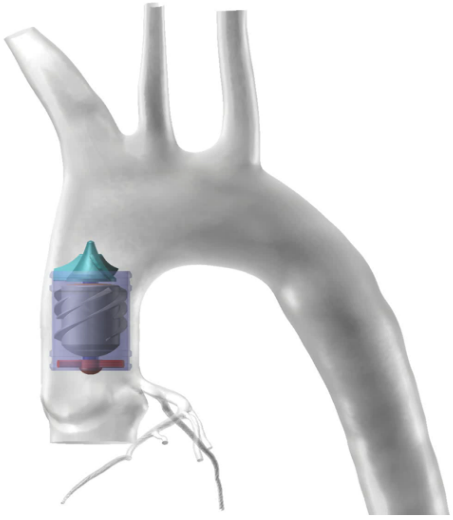
- Simple Picard iteration for linearization
- Lowest order Taylor–Hood element for velocity and pressure
- P^0 element for conformation tensor on a finer grid

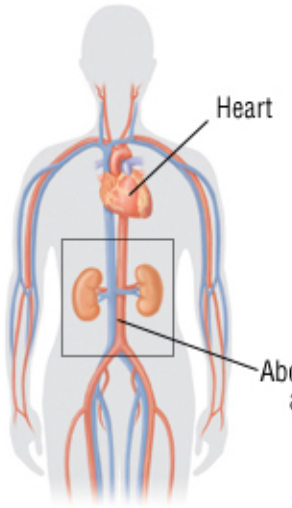
[Lee-Leng-Z. 2016]

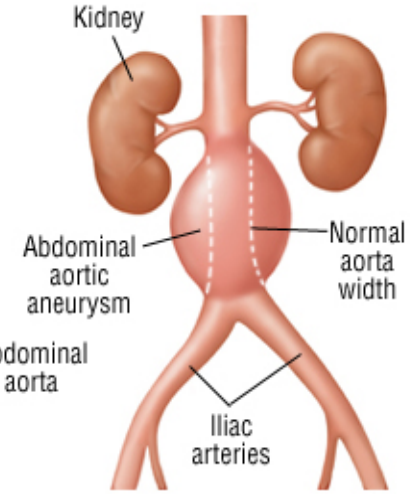
4. Applications in Fluid-Structure Interactions

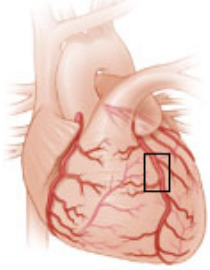
Applications in Cardiovascular Diseases

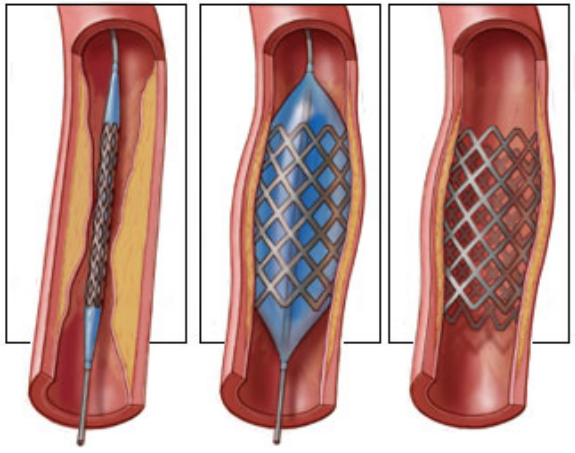


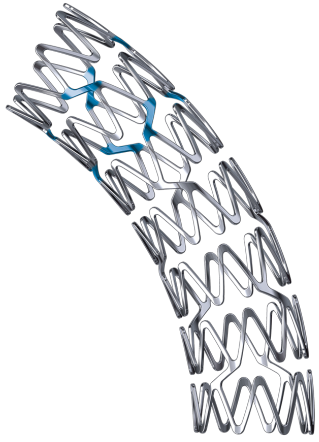












Artificial heart pump, abdominal aortic aneurysm, artery stenosis and dissection, ...

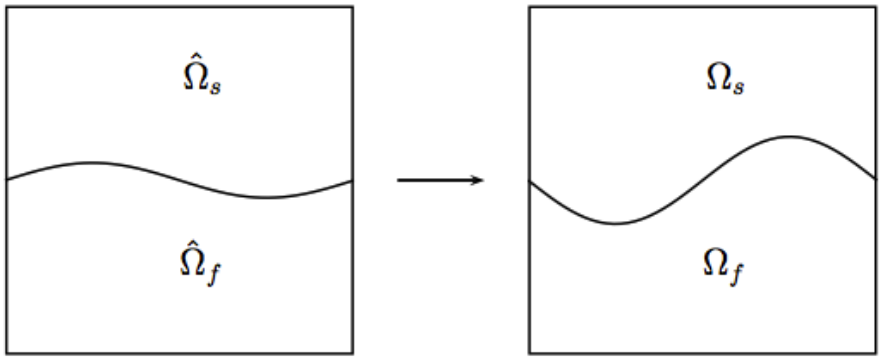
Collaborators on FSI

- Peking Univ 3rd Hospital: Feng Wan
- Kunming Yan'an Hospital: Jinhui Zhang, Huanjun Chen
- LSEC, CAS: Linbo Zhang, Wei Leng, ...
- Penn State Univ: Jinchao Xu, Kai Yang, Wenrui Hao, Shuonan Wu, ...
- Peking Univ: Shihua Gong, Lin Li, Yuyan Chen, ...
- Univ of Nevada, Las Vegas: Pengtao Sun
- Xiangtan Univ: Shi Shu, Haizhuan Yuan, Chunhai Ke, ...
- Sichuan Univ: Xiaoping Xie, Feiteng Huang, ...
- Ohio State Univ: Avner Friedman
- Beijing Univ of Tech: Bin Gao, Yu Chang, ...
- Yunan Co-Creative Center: David Zhu, Zhongjian Zhang, ...

Thanks to NSFC 91430215, 91530323 and Dept of Sci & Tech of Yunan Prov 2014RA071!

Mathematical Models for FSI

A multiphysics problem which studies one or more solid structures (rigid or flexible) interact with an internal or surrounding fluid flow!

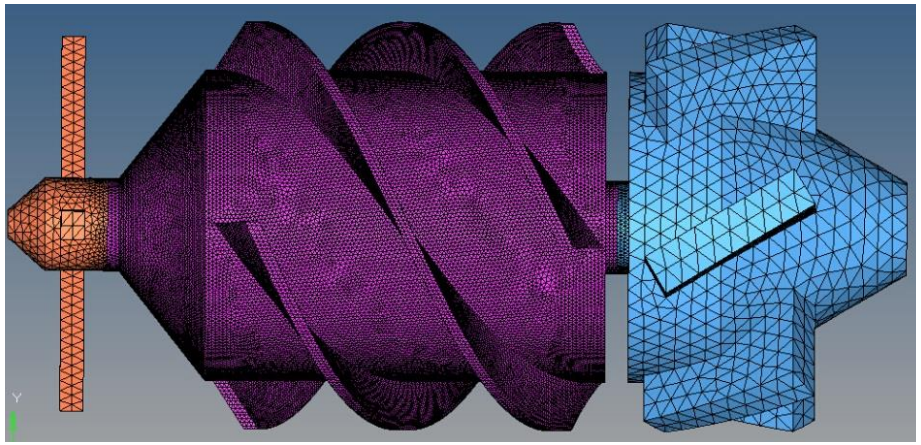


$$\rho D_t v - \nabla \cdot \sigma = g \quad \star \quad v_s = v_f \quad \star \quad \sigma_s n = \sigma_f n$$

- Model: Macroscale vs Mesoscale vs Microscale
- Coupling: Weak vs Strong
- Formulation: Partitioned vs Monolithic
- Coordinate: Eulerian vs Lagrangian or ALE
- Meshing: Conforming vs Non-conforming
- Interface: Tracking vs Capturing

[Richter 2010; Hou-Wang-Layton 2012; Bazilevs-Takizawa-Tezduyar 2013; ...]

Simulating Artificial Heart



Click here to play the FSI movie

A Modified Geometry-Convective Explicit Scheme

Semi-discrete GCE ALE problem:

Find $\left(v_f^{n+1}, \hat{v}_s^{n+1}\right) \in \mathbb{V}^n$ and $p \in \mathbb{Q}^n$ such that $\forall \left(\phi_f, \hat{\phi}_s\right) \in \mathbb{V}^n$ and $\forall q \in \mathbb{Q}^n$,

$$\begin{cases} \frac{1}{k}\left(\rho_f v_f^{n+1}, \phi_f\right)_{\Omega_f} + \left(\sigma_f^{n+1}, \epsilon(\phi_f)\right)_{\Omega_f} + \frac{1}{k}\left(\hat{\rho}_s \hat{v}_s^{n+1}, \hat{\phi}_s\right)_{\hat{\Omega}_s} \\ \qquad + k\left(\tilde{P}_s(\hat{v}_s^{n+1}), \nabla\hat{\phi}_s\right)_{\hat{\Omega}_s} = \left\langle \tilde{g}_f, \phi_f \right\rangle_{\Omega_f} + \left\langle \tilde{g}_s, \hat{\phi}_s \right\rangle_{\hat{\Omega}_s} \\ \qquad \left(\nabla \cdot v_f^{n+1}, q\right)_{\Omega_f} = 0 \end{cases}$$

Define bilinear forms and norms:

$$\begin{aligned} a(v,\phi) &:= k^{-1}(\rho_f v_f, \phi_f)_{\Omega_f} + (\mu_f \epsilon(v_f), \epsilon(\phi_f))_{\Omega_f} \\ &\quad + k^{-1}(\hat{\rho}_s \hat{v}_s, \hat{\phi}_s)_{\hat{\Omega}_s} + k(\mu_s \epsilon(\hat{v}_s), \epsilon(\hat{\phi}_s))_{\hat{\Omega}_s} + k(\lambda_s \nabla \cdot \hat{v}_s, \nabla \cdot \hat{\phi}_s)_{\hat{\Omega}_s} \\ b(v,p) &:= (\nabla \cdot v, p)_{\Omega_f} \\ \|v\|_{\mathbb{V}}^2 &:= a(v,v) + r\|\nabla \cdot v_f\|_{0,\Omega_f}^2 \qquad \|q\|_{\mathbb{Q}}^2 := r^{-1}\|q\|_0^2 \end{aligned}$$

where $r := \max\{1, \mu_f, k^{-1}\rho_f, k^{-1}\hat{\rho}_s, k\mu_s, k\lambda_s\}$.

Well-posedness: $a(\cdot,\cdot)$ is bounded and coercive in $\mathbb{Z}$, $b(\cdot,\cdot)$ is bounded and satisfies the inf-sup condition.

Uniform Stable Discretizations

Finite element discretization:

- Suppose $\mathbb{V}_h \subset \mathbb{V}$ and $\mathbb{Q}_h \subset \mathbb{Q}$ satisfies the inf-sup condition, i.e.,

$$\sup_{v_h \in \mathbb{V}_h} \frac{(\nabla \cdot v_h, q_h)}{\|v_h\|_{1,h}} \gtrsim \|q_h\|_0, \quad q_h \in \mathbb{Q}_h$$

- Denote the kernel spaces as

$$\begin{cases} \mathbb{Z} := \{v \in \mathbb{V} \,:\, \nabla \cdot v = 0 \ \text{ in } \ \Omega\} \\ \mathbb{Z}_h := \{v_h \in \mathbb{V}_h \,:\, (\nabla \cdot v_h, q_h) = 0, \ \forall q_h \in \mathbb{Q}_h\} \end{cases}$$

When is a 'good' Stokes element 'good' for the interface problem?

$$\text{Uniform stability} \iff \mathbb{Z}_h = \{v_h \in \mathbb{V}_h \,:\, \nabla \cdot v_h = 0 \ \text{ in } \Omega_f\}$$

- We mentioned earlier, uniform stability is important for preconditioning
- Standard Stokes element spaces does not satisfies the condition

[Xu-Yang 2015]

Performance of the Block Preconditioners

Use stabilization / equivalent norm [Xie-Xu-Xue 2008; Xu-Yang 2015]

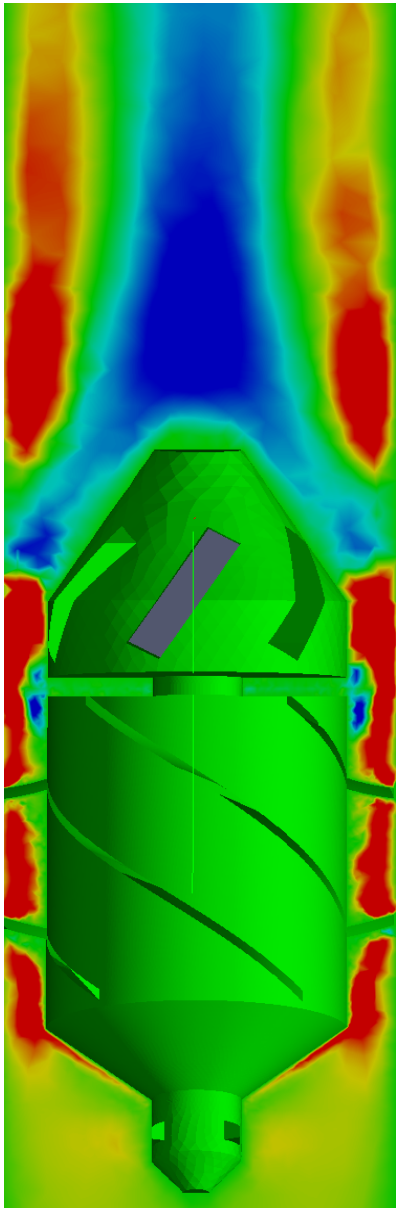
Method	Preconditioner	Stiffness Matrix
M1	$\begin{pmatrix} A_h + rD_h & 0 \\ 0 & \frac{1}{r}M_p \end{pmatrix}^{-1}$	$\begin{pmatrix} A_h + rD_h & B_h^T \\ B_h & 0 \end{pmatrix}$
M2	$\begin{pmatrix} A_h + rD_h^Q & 0 \\ 0 & \frac{1}{r}M_p \end{pmatrix}^{-1}$	$\begin{pmatrix} A_h & B_h^T \\ B_h & 0 \end{pmatrix}$
M3	$\begin{pmatrix} A_h + rD_h' & 0 \\ 0 & \frac{1}{r}M_p \end{pmatrix}^{-1}$	$\begin{pmatrix} A_h + rD_h' & B_h^T \\ B_h & 0 \end{pmatrix}$

Number of iterations for a 2D benchmark [Turek-Hron 2006]

Time stepsize	$k = 0.01$			$k = 0.001$			$k = 0.0001$		
Mesh	M1	M2	M3	M1	M2	M3	M1	M2	M3
1	9	6	11	9	6	11	8	7	11
2	9	6	11	9	7	11	7	7	11
3	9	6	11	8	7	11	9	5	12

Density ratio	$\hat{\rho}_s = \rho_f$			$\hat{\rho}_s = 10\rho_f$			$\hat{\rho}_s = 100\rho_f$		
Mesh	M1	M2	M3	M1	M2	M3	M1	M2	M3
1	15	8	15	9	6	11	7	5	9
2	14	8	15	9	6	11	7	6	9
3	13	8	15	9	6	11	8	6	9

Preliminary Numerical Result



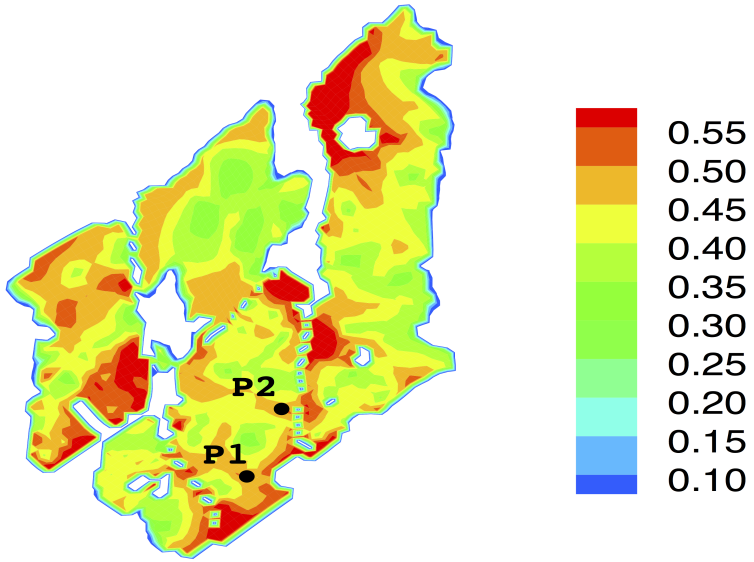
- Fixed flux at inlet
- Fixed angular speed ω of the turbine (7000 rpm)
- Parameters:

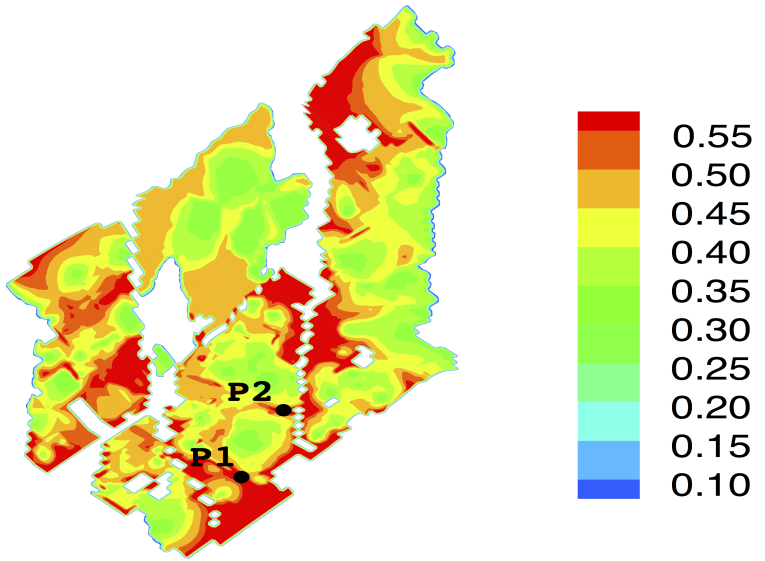
Fluid density	1.057e+3	kg/m^3
Fluid viscosity	3.7e-3	Ns/m^{-2}
Aorta diameter	2	cm
Inflow velocity	0.4	m/s
Reynolds number (aorta)	$\sim$2000	
Reynolds number (turbine)	$\sim$10000	

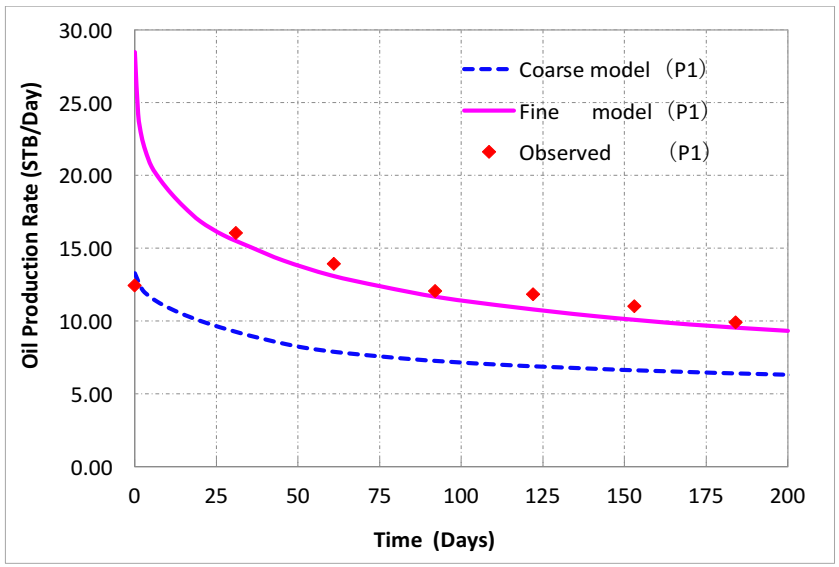
- Experimental pressure difference (6650 Pa)
- Implementation is based on FEniCS, PHG, Gmsh, FASP, PETSc, hypre and METIS
- Number of cores = 1000
- Error is $\approx$ 10% with 14M DOFs

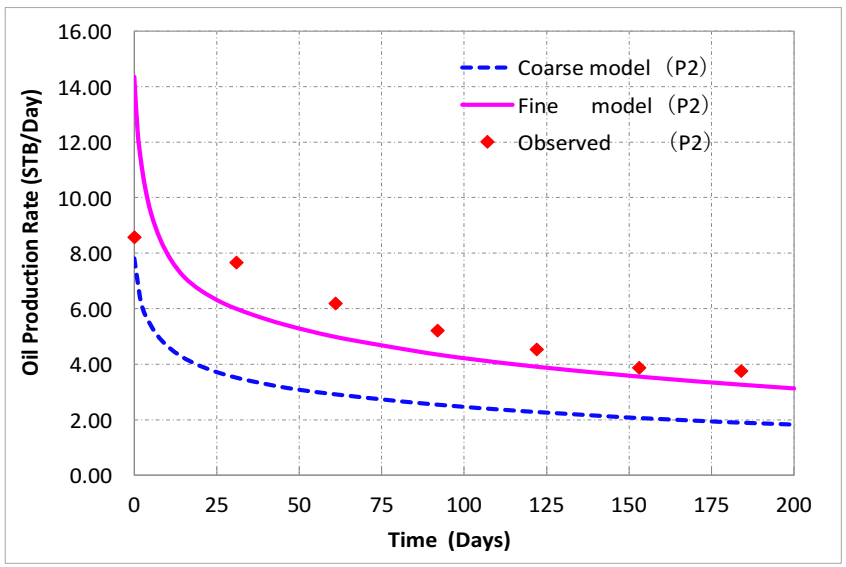
5. Applications in Petroleum Reservoir Simulation

Large-Scale Reservoir Simulation









[Li, Wu, Z., et al., submitted]

Collaborators on PRS

- RIPED, PetroChina: Shuhong Wu, Baohua Wang, Qiaoyun Li, ...
- CNOOC: Xiansong Zhang, Chunyang Lin, ...
- Monix Energy: Wei Liu, ...
- Penn State Univ: Jinchao Xu, Changhe Qiao, Hongxuan Zhang, ...
- Xiangtan Univ: Shi Shu, Chunsheng Feng, ...
- Tufts Univ: Xiaozhe Hu
- SYSU, Guangzhou: Yuesheng Xu, Wenchao Guan, Yongdong Zhang, ...
- Kunming Univ of Sci & Tech: Zheng Li, ...
- Sichuan Univ: Shiquan Zhang

Thanks to CNOOC-2010-ZHKY-ZX-008, NSFC 91130011, PetroChina-12HT10500002654!

Simulating Artificial Heart

Click here to play the SPE10 movie

A Compositional Model

$$\frac{\partial}{\partial t}\Big(\phi \sum_{j=1}^{n_p} x_{ij}\xi_j S_j\Big) + \nabla \cdot \mathrm{F}_i - \sum_{j=1}^{n_p} S_j r_{ij} = Q_i, \qquad i = 1 : n_c$$

$$\mathrm{F}_i = \sum_{j=1}^{n_p} \Big(x_{ij}\xi_j u_j - S_j \mathrm{D}_j \nabla(\xi_j x_{ij})\Big), \qquad i = 1 : n_c$$

$$u_j = \frac{k k_{rj}}{\mu_j}(\nabla P_j - \gamma_j \nabla Z), \qquad j = 1 : n_p$$

$$f_{ij} = f_{i1}, \qquad i = 1 : n_c,\ j = 2 : n_p$$

$$\sum_{j=1}^{n_p} S_j = 1,$$

$$\sum_{i=1}^{n_c} x_{ij} = 1, \qquad j = 1 : n_p$$

[Collins-Nghiem-Li-Grabenstetter 1992; Qiao-Li-Johns-Xu 2014, 2015]

Fully Implicit Discretization

Set of equations and unknowns:

- Primary equations: n_c mass conservation laws and volume balance:

$$V^{\text{fluid}}(P, T, N_1, \ldots, N_{n_c}) = V^{\text{pore}}(P)$$

- Secondary equations: phase equilibrium, density, relative permeability, ...
- Primary unknowns: $\vec{X}_p := (P, N_1, \ldots, N_{n_c})^T$ $\longleftarrow$ One more variables!
- Secondary unknowns: $\vec{X}_s := (x_{11}, \ldots, x_{n_c n_p}, S_1, \ldots, S_{n_p})^T$

Discrete linear equations:

- Update the primary unknowns (Backward Euler + FVM + Newton)

$$\Psi_p := V^{\text{pore}} - V^{\text{fluid}} = 0$$

$$\Psi_i := \frac{N_i^{n+1} - N_i^n}{\Delta t} + \sum_s F_{i,s}^{n+1} + \sum_w Q_{i,k,w}^{n+1} = 0 \qquad i = 1 : n_c$$

- Jacobian matrix $J := \dfrac{d\vec{\Psi}}{d\vec{X}_p} = \dfrac{\partial\vec{\Psi}}{\partial\vec{X}_p} + \dfrac{\partial\vec{\Psi}}{\partial\vec{X}_s}\dfrac{\partial\vec{X}_s}{\partial\vec{X}_p}$ $\longleftarrow$ Expensive to solve!

Abstract Method for Two-Phase Problems

Define auxiliary spaces:

$$\bar{\mathbb{V}} = \mathbb{V} \times \mathbb{V}_P \times \mathbb{V}_S \times \mathbb{V}_{well}$$

Preconditioning algorithm: Given u_0, $Bu_0 := u_4$, where

$$\begin{aligned}
u_1 &= u_0 + \Pi_{well} A_{well}^{-1} \Pi_{well}^* (f - Au_0) \\
u_2 &= u_1 + \Pi_S A_S^{-1} \Pi_S^* (f - Au_1) \\
u_3 &= u_2 + \Pi_P A_P^{-1} \Pi_P^* (f - Au_2) \\
u_4 &= u_3 + S(f - Au_3)
\end{aligned}$$

- Decouple different unknowns effectively
- Form subspaces according to physical properties $\Longrightarrow$ blocks
- Choose appropriate solvers for each subspace
- Example: CPR-type preconditioners

[Hu-Xu-Z., 2013; Qiao-Wu-Xu-Z., 2016]

Convergence and Robustness

No	Name	Properties				Ecl100		HiSim	
		Model	# Total Cells	# Active Cells	Peroid (day)	Newton	Time (min)	Newton	Time (min)
1	SPE10-2	Two-phase	1122000	1094422	2000	——	——	295	41.82
2	SPE9-9k	Black-oil	9000	9000	900	339	0.12	269	0.20
3	SPE1	CO2 flooding	300	300	3656	536	0.04	445	0.08
4	SPE2	Three-phase coning	150	150	900	209	0.01	538	0.14
5	SPE10-3	Black-oil	1122000	1094422	2000	——	——	1462	354.12
6	SPE6	Dual porosity	100	100	7300	306	0.01	322	0.02
7	DPSP	Dual porosity	60984	40294	360	545	2.64	116	0.81
8	SPE7	Horizontal wells	488	488	1500	120	0.01	75	0.02
9	Voliatle	Extended black-oil	2100	2100	0.694			67	0.03
10	Zaoyuan	Field test (black-oil)	417480	143786	10653	3302	105.49	5204	66.20
11	Jidong	Field test (black-oil)	335664	154598	10587	1091	139.69	161	4.41
12	Chengbei	Field test (black-oil)	1646500	585123	2191	1971	155.57	420	28.47
13	Daqing1	Field test (black-oil)	1453248	466913	15616	——	——	5227	338.00
14	Daqing2	Field test (black-oil)	847895	241474	15096	8562	92.46	3072	88.05
15	SPE10-10M	Two-phase (large-scale)	11220000	10944220	2000	——	——	592	962.12
16	SPE9-9M	Black-oil (large-scale)	9000000	9000000	900	——	——	2460	10932.81

Tested by the Research Institute of Petroleum Exploration and Development, PetroChina (2015): Dell E5-2690 v2 CPU@3.0GHz, 200GB DDR3, Windows 7/VS2010/Intel Fortran Compiler 2015, HiSim 2.0, ECL 2012

Strong Scaling Tests on Tianhe-2

Problem	Size	# Nodes	# Processes	Efficiency	Total time (s)	Solver time(s)	# Newton	# Linear
SPE1-refine	75M cells	40	960		888	583.205	76	614
		60	1440	94%	631	425.531	76	631
		80	1920	89%	497	341.551	76	688
		100	2400	68%	520	367.979	87	958
SPE1-refine	150M cells	80	1920		1294	899.789	81	884
		100	2400	118%	878	609.921	82	818
		120	2880	108%	802	566.173	84	860
		160	3840	95%	678	472.916	92	907
		200	4800	84%	614	438.111	100	1095
SPE9-refine	90M cells	32	768		1953	1079.351	133	277
		128	3072	86%	567	307.249	148	525
		256	6144	64%	381	241.309	148	525

SPE1 and SPE9 benchmark problems refined and then tilted. Strong-scaling parallel tests on the Tianhe-2 cluster, Guangzhou: 2nd among Top500 now, 3.12M cores (32K CPUs), 1.408 PB RAM, Rmax 33.86 PFlops, Rpeak 54.90 PFlops, Power 17.8 MW/hr.

[Guan, Qiao, Zhang, Z., et al. 2015]

6. Applications in Optimization Problems

Convex Minimization Problems

Model problem: Consider a convex minimization problem

$$\min_{v \in \mathbb{K}} \mathcal{J}(v)$$

- $\mathcal{J}$ is a convex functional
- $\mathbb{K}$ is a nonempty convex subset of $\mathbb{V}$

Possible Applications

- Obstacle problem
- Contact problem
- Optimal control problem
- American option pricing problem (Brownian motion or Levy process)

Multilevel methods

- SSC and PSC methods [Tai, Wang, Xu, Badea, ...]
- Combining methods like IP with MG [Bank et al. 2003]
- Full approximation scheme [Brandt-Cryer 1983]
- Monotone multigrid method [Kornhuber 1994, 1996]
- Constraint decomposition method [Tai 2003]

Constraint Decomposition Method

Algorithm (SSC)

Let $w^0 = u$

For $i = 1 : m$

$$d_i = \operatorname{argmin} \left\{ \mathcal{J}(w^{i-1} + d_i) \,|\, w^{i-1} + d_i \in \mathbb{K} \text{ and } d_i \in \mathbb{V}_i \right\}$$

Let $w^i = w^{i-1} + d_i$

End For

Algorithm (CDM)

For $k = 0, 1, \ldots,$ till convergence

Decompose $u^k = \sum_{i=1}^{m} u_i$, such that $u_i \in \mathbb{K}_i$; and let $w^0 = u^k$

For $i = 1 : m$

$$w^i = w^{i-1} + \operatorname{argmin}_{d_i} \left\{ \mathcal{J}(w^{i-1} + d_i) \,|\, d_i \in \mathbb{V}_i \text{ and } u_i + d_i \in \mathbb{K}_i \right\}$$

End For

Let $u^{k+1} = w^m$

End For

Convergence Rate – Assumptions

- Discrete feasible set: $\mathbb{K} := \{v \in \mathbb{V} \,|\, v \geq 0\}$
- Decomposition: $\mathbb{K} = \sum_{i=0}^{m} \mathbb{K}_i$ and $\mathbb{K}_i := \{v \in \mathbb{V}_i \,|\, v \geq 0\}$

Assumptions

1. For any $u, v \in \mathbb{K}$, there exist a constant $C_1 > 0$ and decompositions $u = \sum_{i=1}^{m} u_i$ with $u_i \in \mathbb{K}_i$, $v = \sum_{i=1}^{m} v_i$ with $v_i \in \mathbb{K}_i$ such that
$$\left(\sum_{i=1}^{m} \|u_i - v_i\|^2 \right)^{\frac{1}{2}} \leq C_1 \|u - v\| ;$$
2. There exists $C_2 > 0$ such that
$$\sum_{i,j=1}^{m} |\langle \mathcal{J}'(w_{ij} + v_i) - \mathcal{J}'(w_{ij}), \tilde{v}_j \rangle| \leq C_2 \left(\sum_{i=1}^{m} \|v_i\|^2 \right)^{\frac{1}{2}} \left(\sum_{j=1}^{m} \|\tilde{v}_j\|^2 \right)^{\frac{1}{2}},$$
for any $w_{ij} \in \mathbb{V}$, $v_i \in \mathbb{V}_i$, and $\tilde{v}_j \in \mathbb{V}_j$.

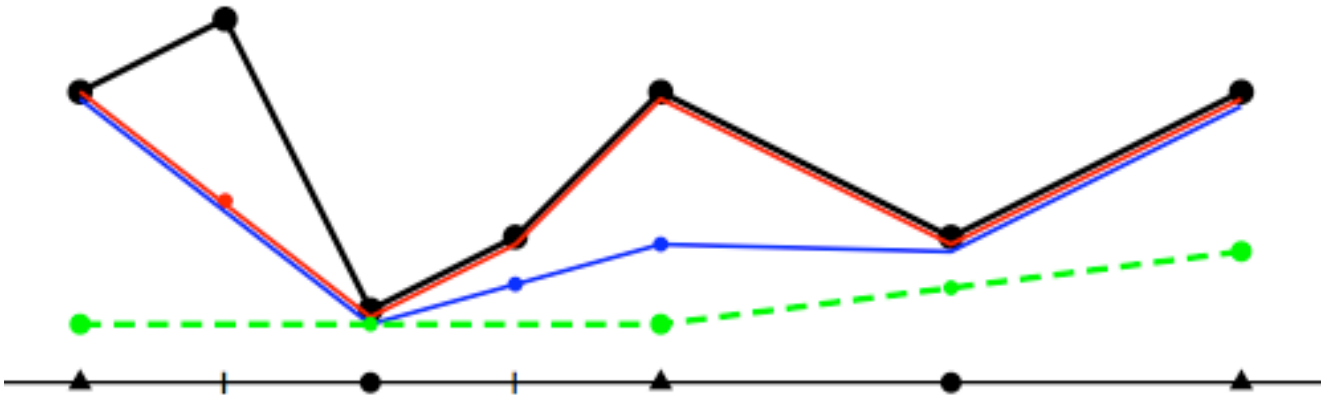
Convergence Rate – Estimate

Theorem (Convergence Rate of CDM, Tai 2003)

$$\frac{\mathcal{J}(w^{i+1}) - \mathcal{J}(u^*)}{\mathcal{J}(w^i) - \mathcal{J}(u^*)} \leq 1 - \frac{1}{(\sqrt{1+C_0} + \sqrt{C_0})^2},$$

where $C_0 = 2C_2 + C_1^2 C_2^2$.

- Linear convergence on uniform grids
- Reduction factor is bounded by a constant depending on C_1 and C_2
- In $\mathbb{R}^2$, reduction factor depends on h mildly



Decomposition on Adaptive Grids

Extension to adaptive grids

- Generalize Tai's decomposition method for adaptive grids [Chen-Nochetto-Z. 2010]
- Newest vertex bisection method and compatible bisections
- Good-for-coarsening and coarsening algorithm [Chen-Z. 2010; Bartels-Schreier 2012; Huang-Xie-Z. 2016]

Numerical Experiments

Adaptive mesh	Degrees of freedom	$h_{\min}$	Reduction factor
1	719	1.563e-2	0.508
2	1199	1.105e-2	0.599
3	2107	7.813e-3	0.660
4	3662	5.524e-3	0.651
5	6560	3.901e-3	0.691
6	11841	2.762e-3	0.701

Table: The reduction factors for the CDM algorithm on adaptively refined meshes. The reduction factor is the ratio of energy error between two consecutive iterations.

7. Error-Resilient Multilevel Iterative Methods

45 Obtain Good Parallel Performance

46 The Moore's Law

47 Communication-Avoiding Algorithms

48 Improve Resilience Using Subspace Corrections

49 Redundant Subspace Corrections

50 Convergence Rate Analysis of RSC

51 Scalability of Redundant Subspace Correction

Obtain Good Parallel Performance

Keys to Good Parallel Performance

extent of parallelism, granularity of partition, locality of computation

The Amdahl's Law

- The parallel speedup is limited by the time needed for sequential portions

$$\text{speedup}(p) = \frac{\text{time}(1)}{\text{time}(p)} = \frac{\text{time}(1)}{\text{Seq} + \text{Par}/p} \leq \frac{\text{time}(1)}{\text{Seq}}$$

- If $1/4$ of the execution time is sequential, then the max speedup is 4!

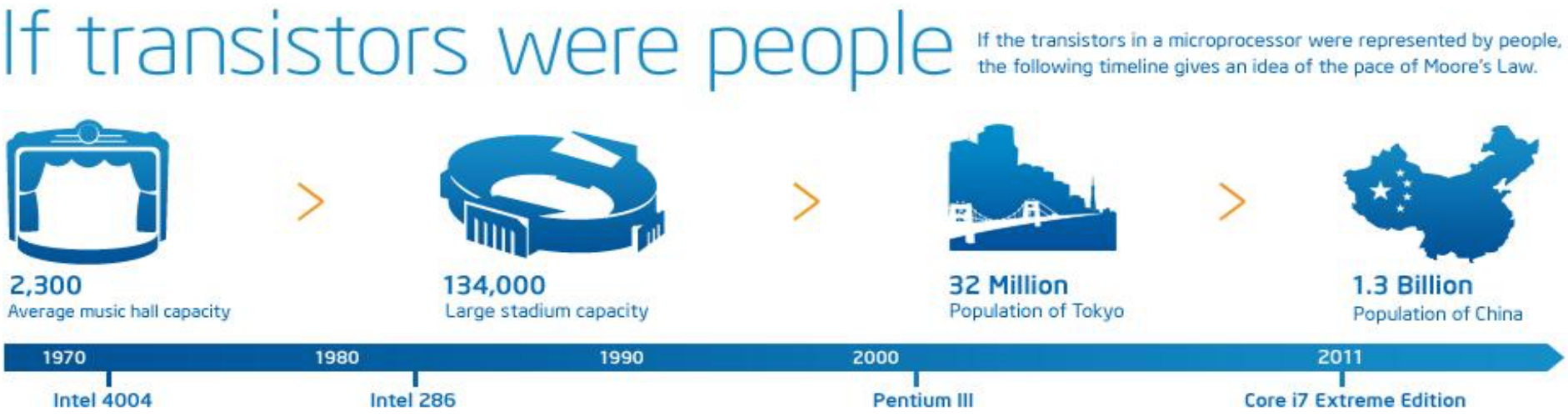
A HPC Paradox (G. Wittum)

- Assumption that algorithm complexity is $E_0 = O(n^q),\ q \geq 1$
- Suppose we want to buy a computer $\alpha \gg 1$ times larger than the old one
- We wish to solve problems of size α times larger than the original
- The new computer then needs computing time proportional to

$$E_1 = O(\alpha^q n^q) = \alpha^q E_0 = \alpha^{q-1} \alpha E_0 \geq \alpha E_0$$

- Weak scalability $\implies$ $E_1 = \alpha E_0$ $\implies$ $q = 1$ (optimal algorithm)

The Moore's Law



Challenges to Keep Up With the Moore's Law:

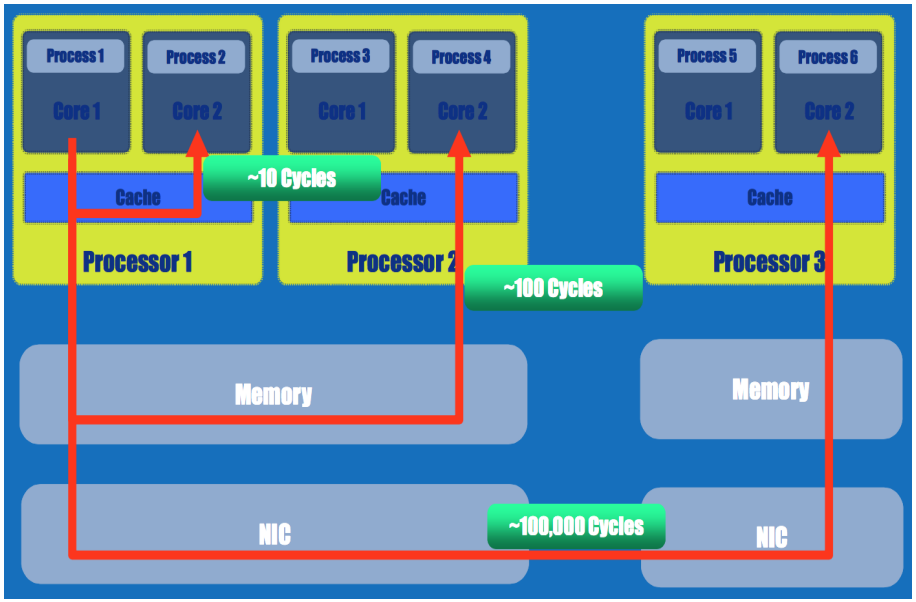
- Instruction-level parallelism (ILP) wall: availability of enough parallel instructions for a multi-core chip
- Power wall: the chip's overall temperature and power consumption

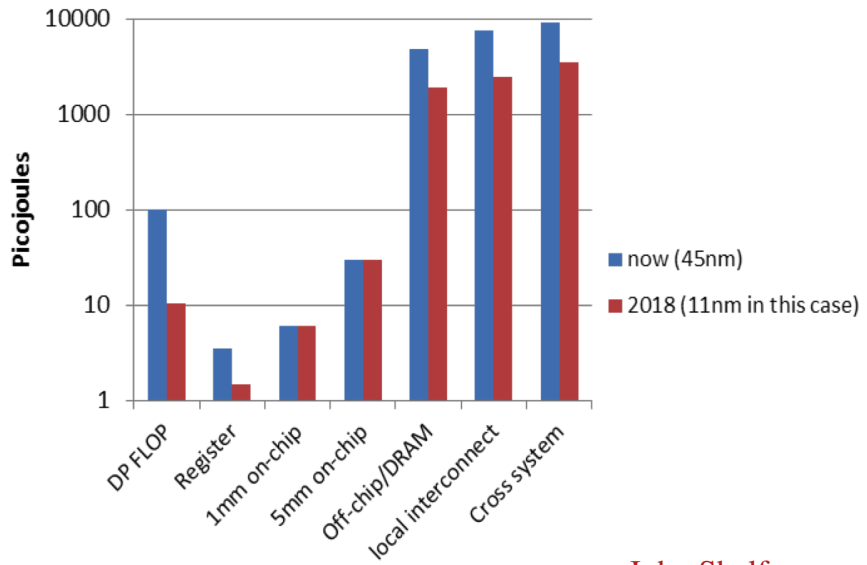
$$\text{Dynamic Power} = K \cdot (\text{Capacitive Load}) \cdot (\text{Voltage})^2 \cdot (\text{Frequency})$$

- Memory wall: bandwidth/latency of the channel b/w CPU and RAM

[Waldrop 2016, Nature]

Communication-Avoiding Algorithms





John Shalf

Floating-point time	$\ll$	1 / Memory bandwidth	$\ll$	Memory latency
59%		23%		5%
Floating-point time	$\ll$	1 / Network bandwidth	$\ll$	Network latency
59%		26%		15%

- Avoid communication (data movement) to save time $\Longrightarrow$ Redesign algorithms
- Linear algebra, LAPACK/ScaLAPACK, ... J. Demmel and collaborators

Improve Resilience Using Subspace Corrections

Improving Mean Time To Failure (MTTF)

- ASCI Q computer (12,288 EV-68 processors) in the LANL experienced 26.1 radiation-induced CPU failures per week [Michalak et al. 2005]
- BlueGene/L (128K processors) experiences one soft error in its L1 cache every 4–6 hours due to radioactive decay [Bronevetsky-Supinski 2008]

Dependability of Computer and Software

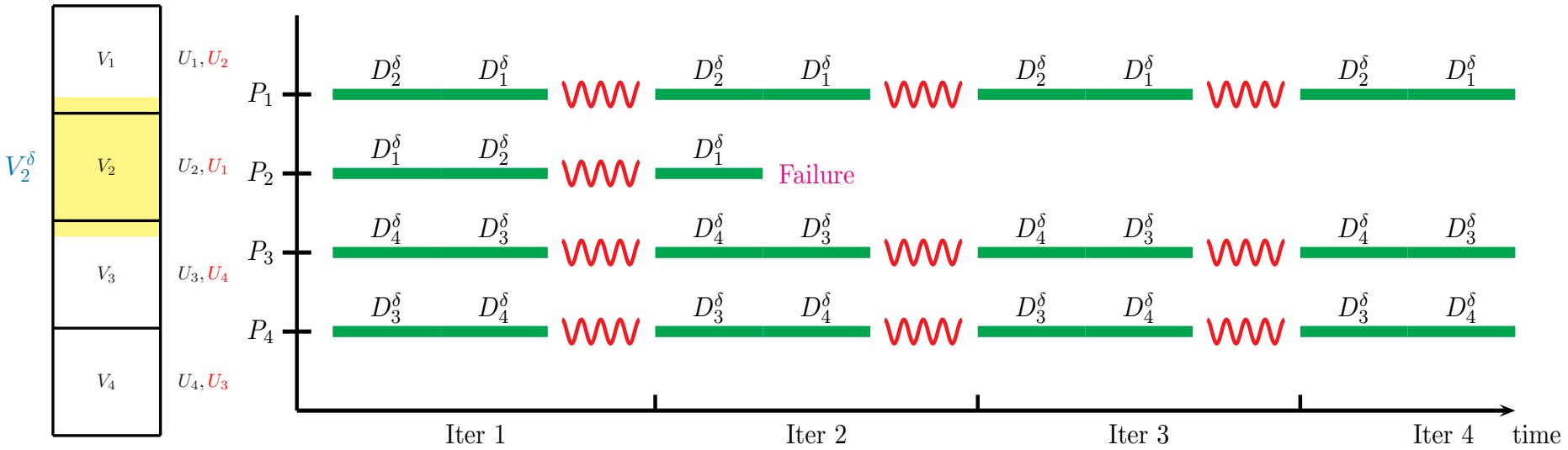
- Properties: availability, reliability, safety, integrity, ...
- Threats: fault, error, failure, ...
- Resilience: forecasting, preventing, removal, ...

Improving Resilience: Redundant Subspace Correction Method

- Maintain convergence when error occurs assuming it is detectable
- Introduce low computational overhead when no error occurs
- Require only small amount of point-to-point communication (locality) and maintain good load balance (granularity)

Redundant Subspace Corrections

Redundant subspace correction (RSC) methods



Some comments

- Light overhead: non-global communication needed
- Resilient to temporary or permanent hardware failures
- Redundancy to maintain convergence when components fail
- Globally parallel and locally successive subspace correction

Convergence Rate Analysis of RSC

Theorem (Xu-Zikatanov 2002)

SSC is convergent if each subspace solvers are convergent. In particular, if the subspace solvers are exact, then

$$\|I - BA\|_A^2 = \Big\|\prod_{i=1}^{J}(I - P_i)\Big\|_A^2 = 1 - \Big(\sup_{\|v\|_A=1}\inf_{\sum_i v_i = v}\sum_{i=1}^{J}\|P_i\sum_{j\geq i} v_i\|_A^2\Big)^{-1}.$$

Theorem (Cui-Xu-Z. 2016)

If an error occurs during computation, the convergence rate of the successive RSC satisfies

$$\|I - B_{\mathrm{SRSC}}A\|_A \leq \|I - B_{\mathrm{SSC}}A\|_A.$$

If there is no error during computation, the convergence rate satisfies that

$$\|I - B_{\mathrm{SRSC}}A\|_A \leq \|I - B_{\mathrm{SSC}}A\|_A \, \|I - \tilde{B}_{\mathrm{SSC}}A\|_A.$$

Scalability of Redundant Subspace Correction

DOFs	#Cores	Error-Free			With Error		
		#Iter	Time	Efficiency	#Iter	Time	Efficiency
1,335,489	16	12	8.09	—	13	8.13	—
2,146,689	32	13	8.64	75.25%	15	8.99	72.68%
4,243,841	64	14	8.91	72.13%	16	9.37	68.93%
10,584,449	128	19	12.87	62.27%	20	13.95	57.73%
16,974,593	256	23	18.01	35.68%	25	19.13	33.76%
33,751,809	512	25	20.90	30.57%	27	26.11	24.59%

Table: Weak-scaling of the PRSC preconditioner for the Poisson equation

#Cores	Standard B_{PSC}			B_{PRSC} Error-Free			B_{PRSC} With Error		
	#Iter	Time	Speedup	#Iter	Time	Speedup	#Iter	Time	Speedup
16	42	162.9	—	21	163.4	—	24	165.3	—
32	48	82.71	1.97	25	82.79	1.97	27	83.35	1.98
64	48	41.14	3.96	26	42.13	3.88	28	43.62	3.79
128	50	20.95	7.78	27	22.66	7.21	29	23.95	6.91
256	51	11.84	13.8	27	13.46	12.1	29	14.03	11.8
512	52	6.91	23.6	28	7.43	21.9	29	7.79	21.2

Table: Strong-scaling of the PRSC for the Poisson equation (16,974,593 DOFs)


Thank You!