Challenging the parallel performance of open-source AMG solvers

Kirill Terekhov[1], **Igor Konshin**[1,2,3]

[1]Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences
[2]Dorodnicyn Computing Centre of the Russian Academy of Sciences
[3]Sechenov University

Plan

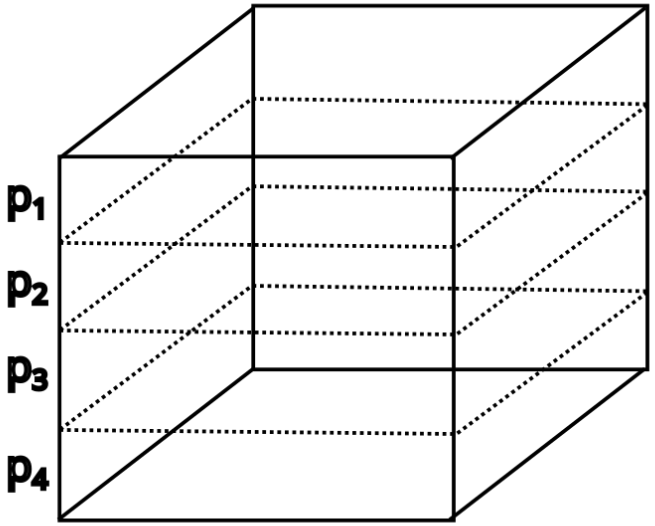
- Goals:
  - faster then PETSc GAMG
  - faster then HYPRE BoomerAMG
  - On symmetric elliptic problems
- Strategy
- Experiments
- Background: Block-AMG methods for Multiphysics FV discretization

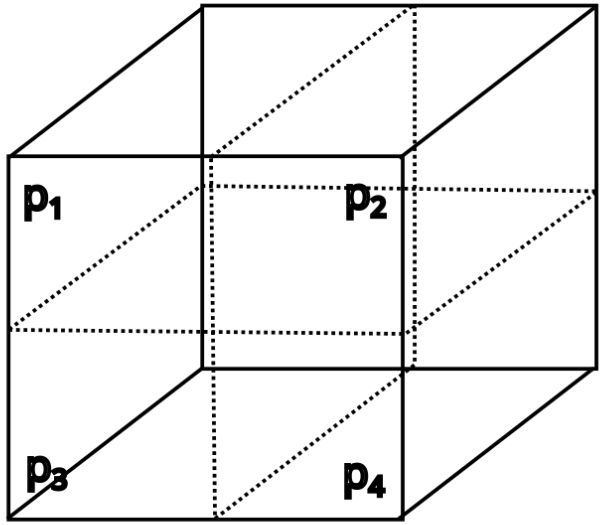
Problem set

- Usual **FD Poisson** problem:
  - NxMxK regular grid, on-the-fly generation
  - Two domain decomposition strategies
  - main testing workhorse
- **FE Poisson** problem:
  - **L2d –** 9-pt stencil on quads
  - **L3d –** 27-pt stencil on hexes
- M3E matrix collection (https://www.m3eweb.it/matrixcollection/)
  - **RTANIS44M** – diffusion in porous media
  - **M20** – mechanical equilibrium
- Type of problems where **AMG** is the most efficient solver

FD Poisson problem

- Two domain decomposition strategies:





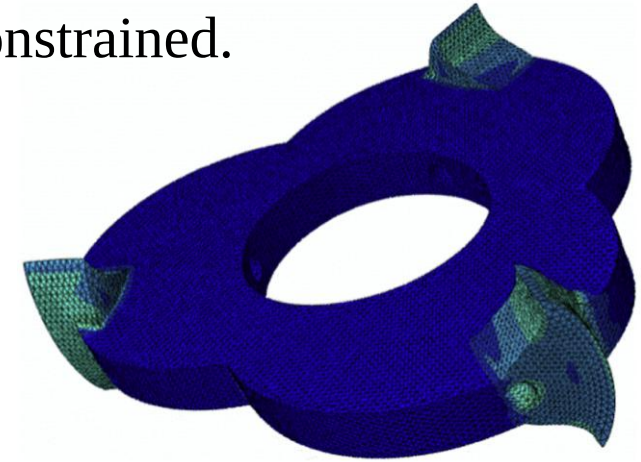
Naïve: cut along matrix rows

Smart: minimize communication volume

- Naïve partitioning gave some hope over rivals
- Other matrices were partitioned with **METIS**

Problem set

- **FE Poisson problem**:
  - L2d **225M** rows, **2.0B** nonzeroes;
  - L3d **100M** rows, **2.6B** nonzeroes.
- **RTANIS44M**: scalar elliptic matrix with **44M** rows and **747M** nonzeroes
  - The mesh derives from a 3D diffusion problem in a porous media.
- **M20**: vector elliptic matrix with **20M** rows and **1.6B** nonzeroes
  - The mesh derives from the 3D mechanical equilibrium of a symmetric machine cutter that is loosely constrained.



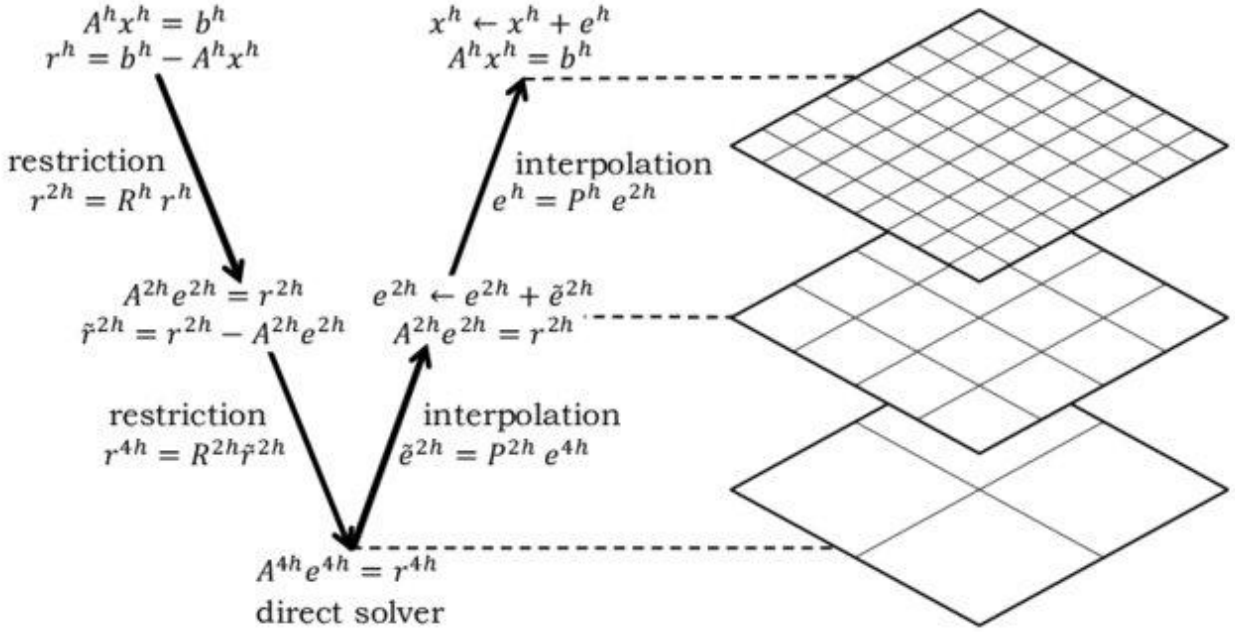
S. Koric et al. / Computers and Structures (2014)

Competition

- **PETSc**:
  - Smoothed aggregation multigrid preconditioner
  - Aggressive coarsening
- **HYPRE**:
  - Classic algebraic multigrid preconditioner
  - No aggressive coarsening is used
- **Common features**: CG iterative method, Chebyshev smoother

Algebraic multigrid method

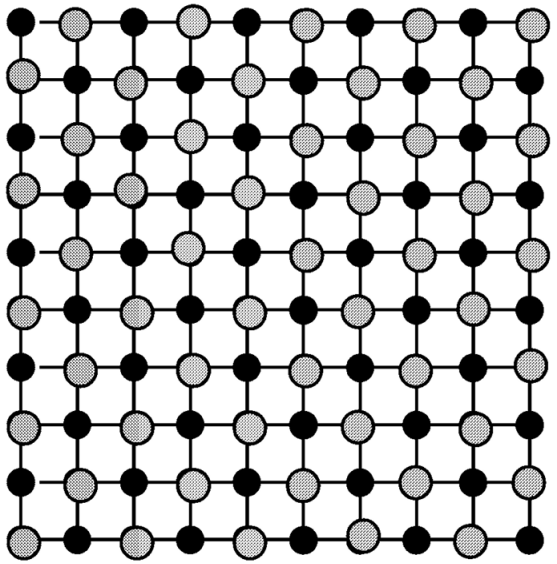
- **Setup** phase:
  - Coarse-fine space splitting.
  - Interpolation and restriction operators.
  - Coarse space computation.
  - Smoother or preconditioner setup.
- **Solve** phase:
  - Smoother application.
  - Matrix-vector multiplication.
- **No grids, only matrices!**



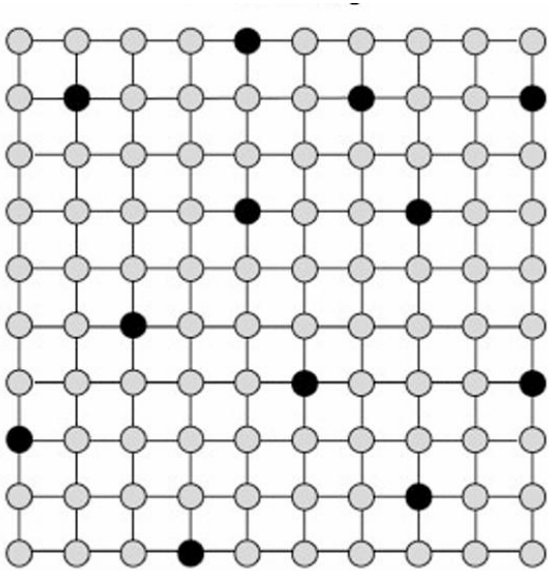
(illustration from internet)

Competition overview

- **Square graph:**
  - Square of the graph of strong connections.
  - **Computationally complex** at the first level.
  - Provides information on neighbours-of-neighbours for **aggressive coarsening**.
  - Allows weighted maximal independent set algorithm - **MWIS(2)**.
- Similar to sparse matrix-matrix product.
- General **MWIS(k)** is a fundamental problem: delivery routs, electric grids…



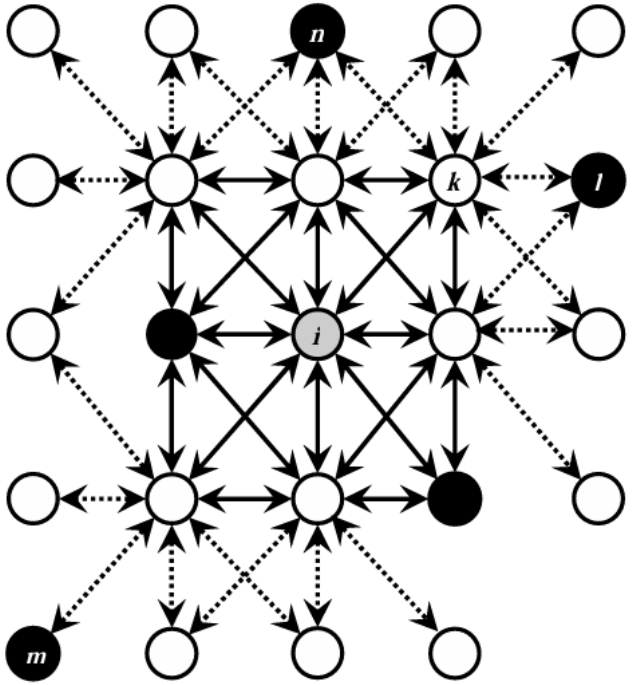
V.E. Henson, U.M. Yang
/ Applied Numerical Mathematics (2002)



U.M. Yang / Numer. Linear Algebra Appl. (2010)

Competition overview

- **Distance-two** interpolation**:**
  - Includes neighbours-of-neighbours into interpolation stencil.
  - **Improves accuracy** of interpolation.
  - Requires **larger overlap** for communication.
  - Significantly **increases** operator complexity:
    - Requires operator truncation.



H. De Sterck et al. / Numer. Linear Algebra Appl. (2007)

Our version of Classic AMG

- **Shared parallel**:
  - Distance-two interpolation
  - Operator truncation
  - Several aggressive coarsening strategies
  - Block version

- **Distributed-hybrid parallel**:
  - Simplest operator construction
  - Unusual L-cycle: visit coarse levels more often

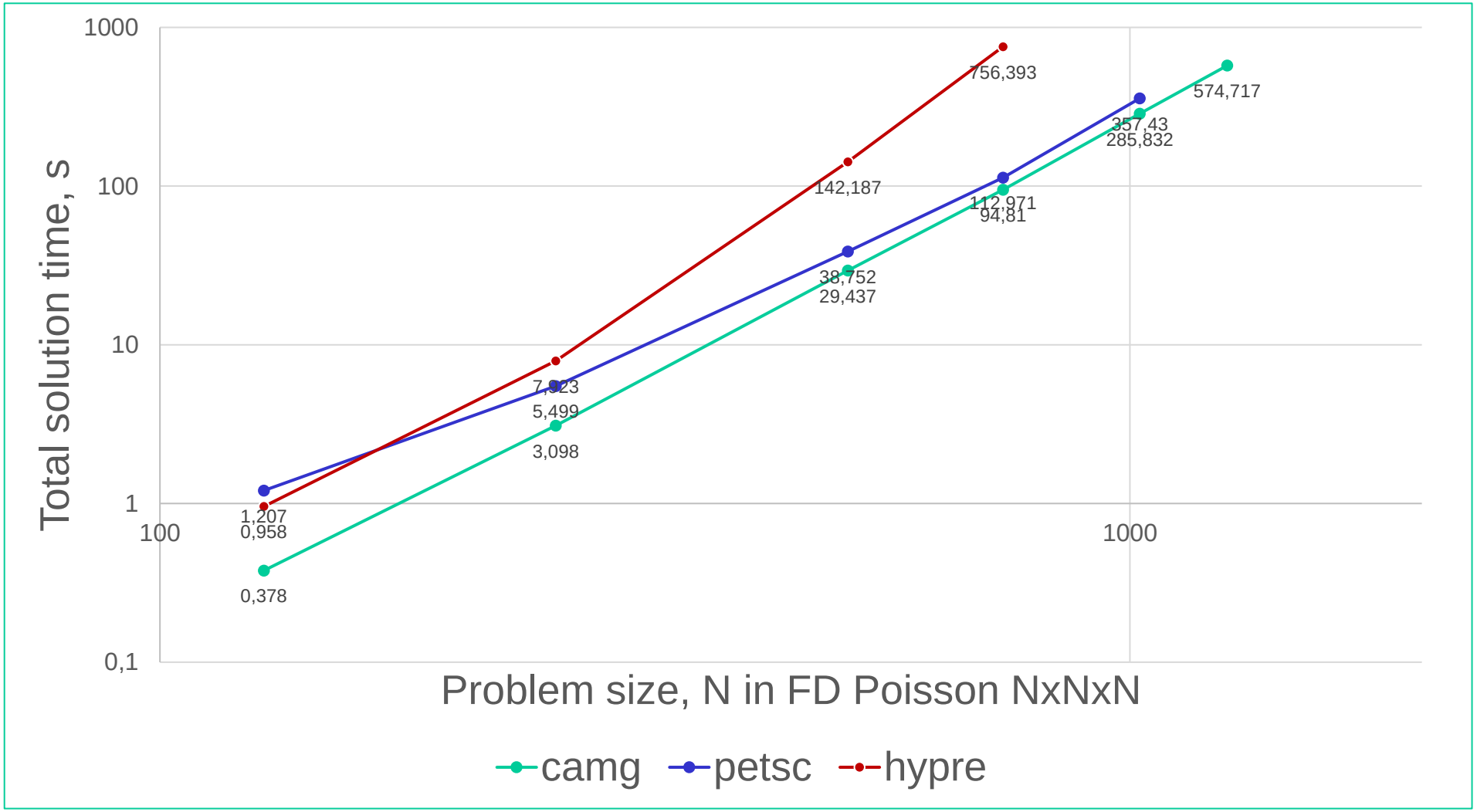
Experiments

- OpenMP version
- MPI version

Shared Parallel Tests

- **Shared version** vs **distributed** PETSc and HYPRE:
  - Reduced tolerance 1e-3 to mimic nonlinear problem solution.
  - **FD Poisson** with **naïve** partitioning: bad communication pattern.
  - Used a single node with 52 cores **vs** 4 nodes with 13 cores (**Sechenov cluster**).

Shared Parallel Tests



Shared Parallel Tests

CAMG

procs	size	total	setup	solve	its
52	128	0,378	0,149	0,228	8
52	256	3,098	1,005	2,092	10
52	512	29,437	8,099	21,338	12
52	740	**94,81**	24,539	70,271	14
52	1024	285,832	67,931	217,901	16
52	1260	574,717	128,454	446,263	17

PETSc

procs	size	total	setup	solve	its
52	128	1,207	0,998	0,209	3
52	256	5,499	4,279	1,219	4
52	512	38,752	30,028	8,723	4
52	740	**112,971**	86,571	26,399	4
52	1024	357,43	263,32	94,11	4

HYPRE

procs	size	total	setup	solve	its
52	128	0,958	0,836	0,122	5
52	256	7,923	6,759	1,165	7
52	512	142,187	126,014	16,173	14
52	740	**756,393**	706,6	49,792	14

Shared Parallel Tests

CAMG

procs	size	total	setup	solve	its
52	128	0,378	0,149	0,228	8
52	256	3,098	1,005	2,092	10
52	512	29,437	8,099	21,338	12
52	740	**94,81**	24,539	70,271	14
52	1024	285,832	67,931	217,901	16
52	1260	574,717	128,454	446,263	17

PETSc

procs	size	total	setup	solve	its
52	128	1,207	0,998	0,209	3
52	256	5,499	4,279	1,219	4
52	512	38,752	30,028	8,723	4
52	740	**112,971**	86,571	26,399	4
52	1024	357,43	263,32	94,11	4

HYPRE

procs	size	total	setup	solve	its
52	128	0,958	0,836	0,122	5
52	256	7,923	6,759	1,165	7
52	512	142,187	126,014	16,173	14
52	740	**756,393**	706,6	49,792	14

Shared Parallel Tests

- **Prelimenary conclusions:**
  - **We are faster!**
  - **Confusions**:
    - PETSc faster then HYPRE
    - HYPRE **badly** scales with system size, but faster iteration
    - PETSc demonstrates textbook efficiency for iteration count
  - Next steps:
    - Reproduce textbook efficiency $O(N)$.
    - Correct matrix partitioning.

Shared Parallel Tests

- Reproducing **textbook efficiency** $O(N)$:
  - **Distance-two** interpolation → **larger** operator complexity → operator truncation
  - Metrics $Cop = \frac{\sum \mathrm{nnz}(A_k)}{nnz(A)}$, $Csize = \frac{\sum size(A_k)}{size(A)}$, $Camg = \frac{\sum nnz(R_k, A_k, P_k)}{nnz(A)}$
  - Gauss-Seidel smoother, tolerance 1e-7
  - Original operator construction:

size	its	lvl	Cop	Csize	Camg
32	**12**	4	2,499	1,431	2,593
64	**14**	5	2,586	1,425	2,601
128	**18**	7	2,623	1,42	2,599
256	**22**	9	2,65	1,42	2,625
512	**28**	11	2,659	1,419	2,633

  - Following H. De Sterck et al. / Numer. Linear Algebra Appl. (2007)

Shared Parallel Tests

- Reproducing textbook efficiency:
  - **Distance-two** interpolation ($Cop \sim 2.65 \rightarrow \sim 4.65$):

size	**its**	lvl	Cop	Csize	Camg
32	**7**	3	4,21	1,377	5,666
64	**7**	4	4,455	1,369	5,894
128	**7**	5	4,567	1,363	6,016
256	**7**	6	4,642	1,363	6,102
512	**7**	7	4,669	1,362	6,132

  - Choose connections by **influence** ($Cop \sim 4.65 \rightarrow \sim 3.8$)

size	**its**	lvl	Cop	Csize	Camg
32	**9**	3	3,508	1,384	4,445
64	**9**	4	3,67	1,377	4,437
128	**9**	5	3,756	1,37	4,51
256	**9**	7	3,806	1,37	4,564
512	**9**	8	3,828	1,369	4,589

Shared Parallel Tests

- Reproducing textbook efficiency:
  - Weight **truncation**, only 5 largest connections ($Cop \sim 3.8 \rightarrow \sim 3.2$):

size	**its**	lvl	Cop	Csize	Camg
32	**9**	3	3,063	1,385	3,809
64	**9**	4	3,165	1,379	3,701
128	**9**	6	3,209	1,372	3,708
256	**10**	7	3,236	1,372	3,733
512	**10**	9	3,247	1,371	3,743

  - Aggressive coarsening using MWIS on **square graph** ($Cop \sim 3.2 \rightarrow \sim 1.7$)

size	**its**	lvl	Cop	Csize	Camg
32	**12**	3	1,477	1,129	1,477
64	**12**	4	1,643	1,123	1,458
128	**12**	5	1,669	1,120	1,482
256	**13**	5	1,683	1,119	1,496
512	**13**	6	1,689	1,119	1,502

Shared Parallel Tests

- **Conclusions:**
  - Textbook efficiency is **possible**.
    - Requires **distance-two** interpolation.
    - Leads to **aggressive** coarsening.
  - But technologically rather **complex**.

- **Distributed** implementation is left for the future works…

Distributed Parallel Tests

- **FD Poisson** with **smart** partitioning (**Sechenov cluster**):
  - Tolerances 1e-7.
  - Competition is much tighter.

PETSc

procs	size	total	setup	solve	its	lvl
1040	128	0,554	0,516	0,0388	3	5
1040	256	0,689	0,606	0,0833	4	6
1040	512	2,087	1,655	0,433	4	6
1040	1024	15,172	11,467	3,705	5	7
104	128	0,639	0,582	0,057	3	5
104	256	2,354	1,83	0,523	4	6
104	512	16,783	12,725	4,057	4	6
104	1024	142,864	105,152	37,712	5	7

  - PETSc demonstrate good scaling in both processor count and problem size.

Distributed Parallel Tests

- **FD Poisson** with **smart** partitioning (**Sechenov cluster**):
  - Tolerances 1e-7.
  - Competition is much tighter.

HYPRE

procs	size	total	setup	solve	its	lvl
1040	128	0,475	0,417	0,0581	4	7
1040	256	0,455	0,325	0,131	5	8
1040	512	2,443	1,449	0,994	7	9
1040	1024	24,298	14,673	9,625	9	10
104	128	0,38	0,294	0,0863	4	7
104	256	2,751	1,807	0,944	5	8
104	512	**31,75**	21,679	10,071	7	9
104	1024	**502,094**	411,942	90,151	8	10

  - HYPRE shows **bad scaling** (but better) with problem size but **very good** scaling with processor count.

Distributed Parallel Tests

- Poisson with **smart** partitioning (**Sechenov cluster**):
  - Tolerances 1e-7.
  - Competition is much tighter.

CAMG V-cycle

procs	size	total	setup	solve	its	lvl
1040	128	0,294	0,254	0,0404	8	7
1040	256	0,567	0,44	0,127	10	9
1040	512	2,055	0,971	1,084	13	11
1040	1024	15,752	5,023	10,729	16	14
104	128	0,256	0,138	0,118	8	7
104	256	1,665	0,646	1,02	10	9
104	512	15,435	4,322	11,113	13	11
104	1024	143,564	33,368	110,196	16	13

  - Performance is a tight match to PETSc, a bit **slower**, we have much faster setup but much slower iterations.

Distributed Parallel Tests

PETSc

procs	size	total	setup	solve	its	lvl
1040	128	0,554	0,516	0,0388	3	5
1040	256	0,689	0,606	0,0833	4	6
1040	512	2,087	1,655	0,433	4	6
1040	1024	**15,172**	11,467	3,705	5	7
104	128	0,639	0,582	0,057	3	5
104	256	2,354	1,83	0,523	4	6
104	512	16,783	12,725	4,057	4	6
104	1024	**142,864**	105,152	37,712	5	7

CAMG V-cycle

procs	size	total	setup	solve	its	lvl
1040	128	0,294	0,254	0,0404	8	7
1040	256	0,567	0,44	0,127	10	9
1040	512	2,055	0,971	1,084	13	11
1040	1024	**15,752**	5,023	10,729	16	14
104	128	0,256	0,138	0,118	8	7
104	256	1,665	0,646	1,02	10	9
104	512	15,435	4,322	11,113	13	11
104	1024	**143,564**	33,368	110,196	16	13

Distributed Parallel Tests

- Poisson with **smart** partitioning (**Sechenov cluster**):
  - Tolerances 1e-7.
  - Competition is much tighter.

CAMG L-cycle

procs	size	total	setup	solve	its	lvl
1040	128	0,306	0,263	0,0428	8	7
1040	256	0,655	0,507	0,148	9	9
1040	512	1,89	0,961	0,929	9	11
1040	1024	13,783	4,981	8,801	9	14
104	128	0,262	0,14	0,121	8	7
104	256	1,599	0,643	0,955	9	9
104	512	12,461	4,319	8,141	9	11
104	1024	99,189	33,272	65,917	9	13

  - Switch to L-cycle, less expensive then F-cycle: regain **textbook efficiency** and solve faster then PETSc!

Distributed Parallel Tests

PETSc

procs	size	total	setup	solve	its	lvl
1040	128	0,554	0,516	0,0388	3	5
1040	256	0,689	0,606	0,0833	4	6
1040	512	2,087	1,655	0,433	4	6
1040	1024	**15,172**	11,467	3,705	5	7
104	128	0,639	0,582	0,057	3	5
104	256	2,354	1,83	0,523	4	6
104	512	16,783	12,725	4,057	4	6
104	1024	**142,864**	105,152	37,712	5	7

CAMG L-cycle

procs	size	total	setup	solve	its	lvl
1040	128	0,306	0,263	0,0428	8	7
1040	256	0,655	0,507	0,148	9	9
1040	512	1,89	0,961	0,929	9	11
1040	1024	**13,783**	4,981	8,801	9	14
104	128	0,262	0,14	0,121	8	7
104	256	1,599	0,643	0,955	9	9
104	512	12,461	4,319	8,141	9	11
104	1024	**99,189**	33,272	65,917	9	13

Distributed Parallel Tests

- Poisson with **smart** partitioning (**Lomonosov supercomputer**):
  - Tolerances 1e-7.
  - Competition is much tighter.

CAMG L-cycle

procs	size	total
700	64	**0,0658**
700	128	0,1388
700	256	0,4497
700	512	3,3142
700	1024	28,9715

PETSc

procs	size	total
700	64	0,0935
700	128	0,1332
700	256	**0,4093**
700	512	**2,9962**
700	1024	**23,4107**

HYPRE

procs	size	total
700	64	0,0675
700	128	**0,1039**
700	256	0,4529
700	512	**3,8693**
700	1024	**39,284**

- **Lost** to PETSc but **won** HYPRE (still has **scalability issues** with problem size)

Distributed Parallel Tests

- **FD Poisson** with **smart** partitioning:
  - **We are faster!** (**Sechenov cluster**)
  - **We have lost! (Lomonosov supercomputer)**
  - But with a very specific **L-cycle**.
  - But the competition is very **tight**.
  - Some room for easy improving:
    - We use Gershgorin circles for eigenvalue estimation in Chebyshev smoother.
    - PETSc uses its internal logic
    - HYPRE improves estimation by several iterations.
  - **Confusion** regarding HYPRE results remains.
- Next steps:
  - more sophisticated systems.

Distributed Parallel Tests

- **L2d**: 9-pt FE Poisson
  - **Sechenov cluster**:

solver	Procs	total	setup	solve	its	lvl	Cop	Remark
petsc	1024	9,6378	3,1696	6,4681	40	7	1,1366	
camg	**1024**	**3,5226**	**1,1584**	**2,3642**	**24**	**11**	**0,7651**	**L, G-S**
camg	1024	3,6431	1,0425	2,6006	19	11	0,7651	L, Cheb
Camg	1024	4,9394	1,2975	3,6419	49	11	0,7651	V, G-S
Camg	1024	5,3371	1,0193	4,3178	43	11	0,7651	V, Cheb

  - **Lomonosov supercomputer:**

CAMG

procs	total
140	23,809
350	**6,015**
700	**3,189**

PETSc

procs	total
140	36,819
350	17,865
700	7,401

HYPRE

procs	total
140	**16,602**
350	6,554
700	3,287

Distributed Parallel Tests

- **L3d**: 27-pt FE Poisson
  - **Sechenov cluster**:

solver	procs	total	setup	solve	its	lvl	Cop	Remark
petsc	1024	8,8402	3,6709	5,1692	31	5	1,0437	
camg	1024	3,1336	1,3161	1,8175	20	9	0,3427	L, G-S
camg	**1024**	**2,6068**	**1,0667**	**1,5401**	**16**	**9**	**0,3427**	**L, Cheb**
camg	1024	3,4667	1,2868	2,1799	26	9	0,3427	V, G-S
camg	1024	3,1398	1,0351	2,1047	25	9	0,3427	V, Cheb

  - **Lomonosov supercomputer:**

CAMG

procs	total
140	31,529
350	**4,594**
700	**2,719**

PETSc

procs	total
140	32,276
350	12,559
700	6,473

HYPRE

procs	total
140	**20,083**
350	8,353
700	4,297

Distributed Parallel Tests

- **RTANIS44M**: anisotropic diffusion problem
  - **Sechenov cluster**:

solver	procs	total	setup	solve	its	lvl	Cop	Remark
petsc	1024	12,7158	2,4659	10,2496	343	5	1,1122	
camg	1024	6,2591	1,4848	4,7742	20	9	1,2834	L, G-S
camg	**1024**	**3,9479**	**1,0935**	**2,8543**	**47**	**9**	**1,2834**	**L, Cheb**
camg	1024	9,7298	1,5437	8,1861	135	9	1,2834	V, G-S
camg	1024	6,8109	1,1884	5,6224	132	9	1,2834	V, Cheb

  - **Lomonosov supercomputer:**

CAMG

procs	total
140	**17,412**
350	**5,971**
700	**3,559**

PETSc

procs	total
140	23,6454
350	8,88162
700	4,11084

HYPRE

procs	total
140	29,7622
350	11,964
700	5,7491

Distributed Parallel Tests

- **M20**: mechanical equilibrium
  - **Sechenov cluster**:

solver	procs	total	setup	solve	its	lvl	Cop	Remark
petsc	1024	15,6776	2,0417	13,6358	343	4	1,0348	
camg	1024	16,2361	2,1251	14,1111	218	8	0,6822	L, G-S
camg	**1024**	**10,4295**	**1,1159**	**9,3135**	**161**	**8**	**0,6822**	**L, Cheb**
camg	1024	22,8045	1,9666	20,8379	341	8	0,6822	V, G-S
camg	1024	19,0869	1,0686	18,0182	323	8	0,6822	V, Cheb

  - **Lomonosov supercomputer:**

CAMG

procs	total
140	65,238
350	**22,670**
700	**11,933**

PETSc

procs	total
140	**48,462**
350	30,977
700	14,691

HYPRE

procs	total
140	117,426
350	29,316
700	14,067

Distributed Parallel Tests

- More sophisticated systems:
  - **We are faster!**
  - But with a very specific **L-cycle** and Chebyshev iterative method.

- **Future work**:
  - Distributed distance-two interpolation with aggressive coarsening.
  - Block version for coupled Multiphysics systems.
    - I Konshin, K Terekhov: **Adaptive block algebraic multigrid method for multiphysics problems**, Computational Mathematics and Mathematical Physics 2025

Thank you for your attention

Contacts:

- kirill.terehov@gmail.com
- igor.konshin@gmail.com

Supported by:

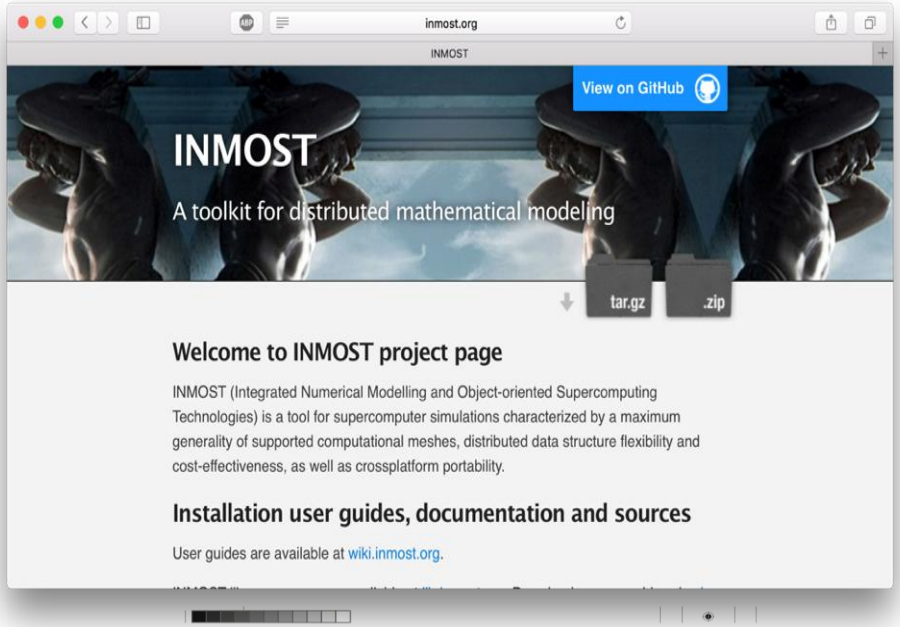
Russian Science Foundation 21-71-20024-П

Parallel computations:

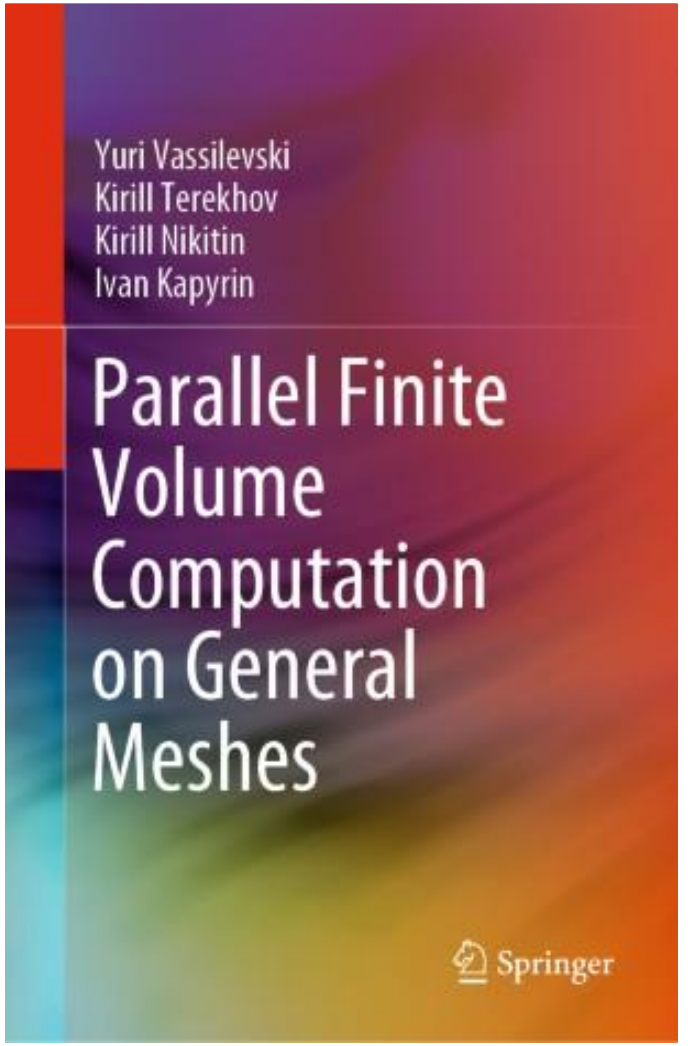
Lomonosov-2 Moscow State University,
Sechenov University

Background

- Parallel frameworks



Websites:
www.inmost.org
www.inmost.ru



More then **20** articles

General Concept for Differential Equations

System of PDE equations:

$$\frac{\partial \tau(q)}{\partial t} + \mathbf{div}\big(\mathcal{A}(q)\big) = \mathcal{R}(q),$$

Where

- q is $N \times 1$ vector of unknowns of the system,
- $\tau(q)$ corresponds to the accumulation,
- $\mathcal{R}(q)$ represents **body forces** and **reactions** – discretized with matrix-weighted Euler method:
  - I. Butakov, K. Terekhov. **Two Methods for the Implicit Integration of Stiff Reaction Systems**. CMAM, 2022.
- $\mathcal{A}(q)$ represents **conservative forces** – addressed by the general finite volume framework:
  - K.Terekhov. **General finite-volume framework for saddle-point problems of various physics.** RJNAMM, 2021

Ultimate goal: automatic **collocated** finite-volume discretization for a given system.

Complications: inf-sup condition, convective instability and other problems…

We get a system with $N \times N$ **blocks**. At the core we get a **symmetric quasi-definite** system.

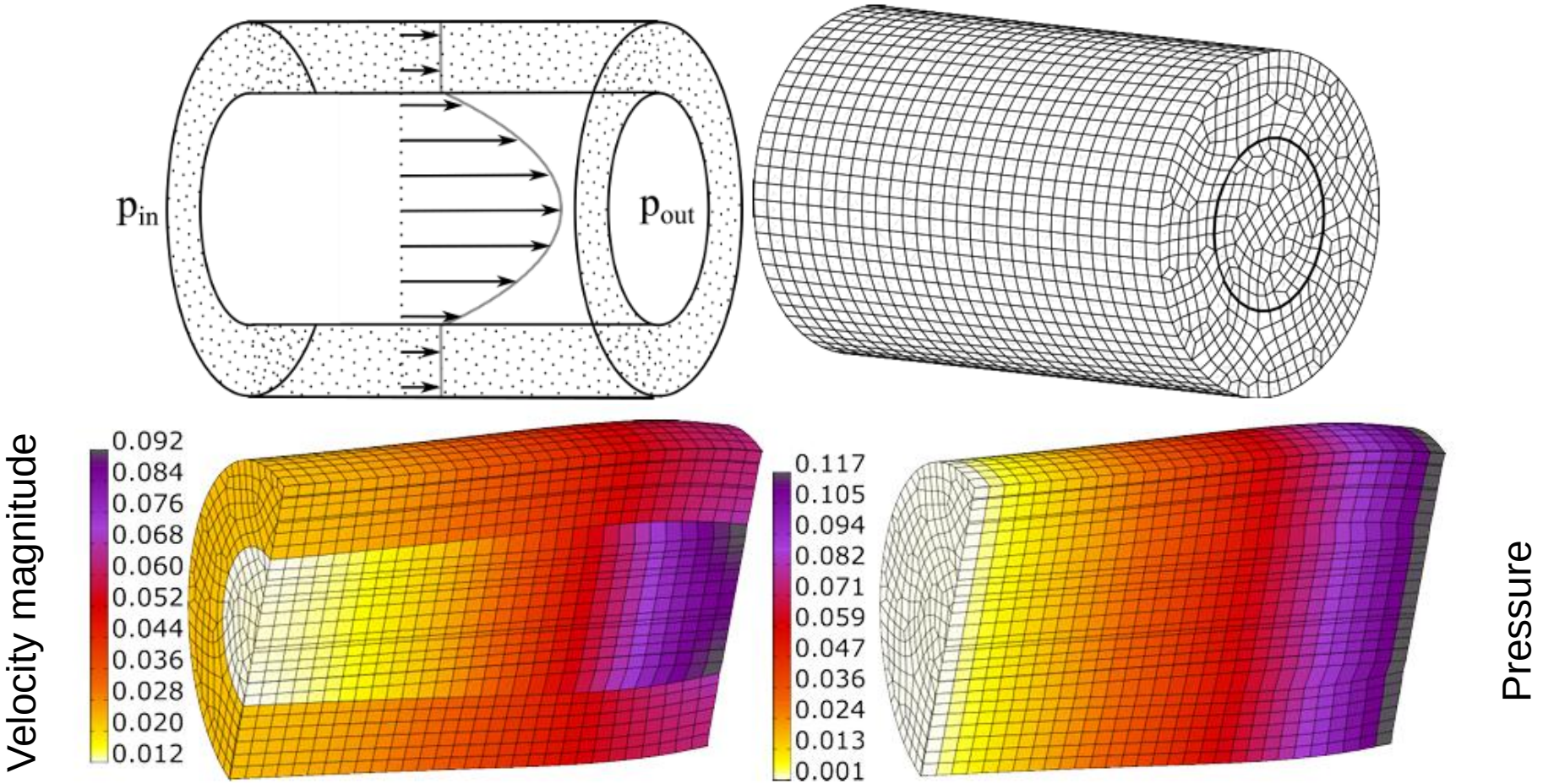
Solved with block adaptive AMG method!

Publications on FV

- K. Terekhov, B. Mallison, and H. Tchelepi. **Cell-centered nonlinear finite-volume methods for the heterogeneous anisotropic diffusion problem.** Journal of Computational Physics, 2017.
- K. Terekhov, and Yu. Vassilevski. **Finite volume method for coupled subsurface flow problems, I: Darcy problem.** Journal of Computational Physics, 2019
- K. Terekhov, and H. Tchelepi. **Cell-centered finite-volume method for elastic deformation of heterogeneous media with full-tensor properties.** Journal of Computational and Applied Mathematics, 2020
- K. Terekhov. **Cell-centered finite-volume method for heterogeneous anisotropic poromechanics problem.** Journal of Computational and Applied Mathematics, 2020
- K. Terekhov. **Collocated Finite-Volume Method for the Incompressible Navier-Stokes Problem**, Journal of Numerical Mathematics, 2020
- Yu. Vassilevski, K. Terekhov, K. Nikitin, I. Kapyrin. **Parallel finite volume computation on general meshes**, Springer Book, 2020
- K. Terekhov. **Multi-physics flux coupling for hydraulic fracturing modelling within INMOST platform.** Russian Journal of Numerical Analysis and Mathematical Modelling, 2020
- K. Terekhov. **Fully-Implicit Collocated Finite-Volume Method for the Unsteady Incompressible Navier-Stokes Problem**, Lecture Notes in Computational Science and Engineering, 2021
- K. Terekhov, and Yu. Vassilevski. **Finite volume method for coupled subsurface flow problems, II: Poroelasticity.** Journal of Computational Physics, 2022
- K. Terekhov **Pressure boundary conditions in the collocated finite-volume method for the steady Navier–Stokes equations.** Computational Mathematics and Mathematical Physics, 2022
- I.. Butakov and K. Terekhov **Two Methods for the Implicit Integration of Stiff Reaction Systems.** Computational Methods in Applied Mathematics, 2022
- K. Terekhov, I. Butakov., A. Danilov, Yu. Vassilevski, **Dynamic adaptive moving mesh finite-volume method for the blood flow and coagulation modeling.** *International Journal for Numerical Methods in Biomedical Engineering*, e3731, 2023
- K. Terekhov. **General finite-volume framework for saddle-point problems of various physics.** Russian Journal of Numerical Analysis and Mathematical Modelling, 2021
- I Konshin, K Terekhov: **Adaptive block algebraic multigrid method for multiphysics problems**, Computational Mathematics and Mathematical Physics 2025

Multiphysics problems

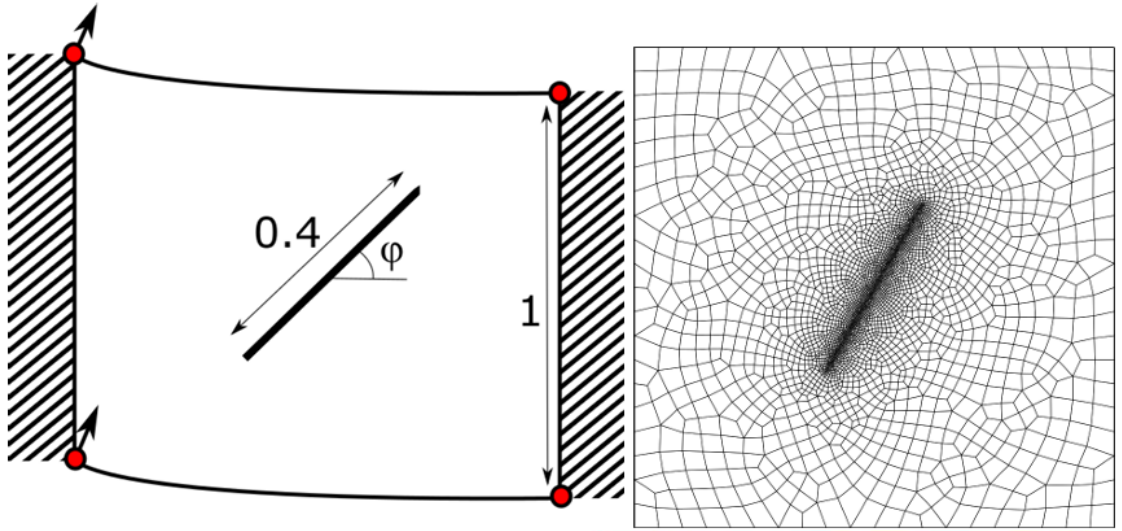
- Navier-Stokes (4x4 block)/Poroelasticity (4x4 block) with Beavers-Joseph conditions

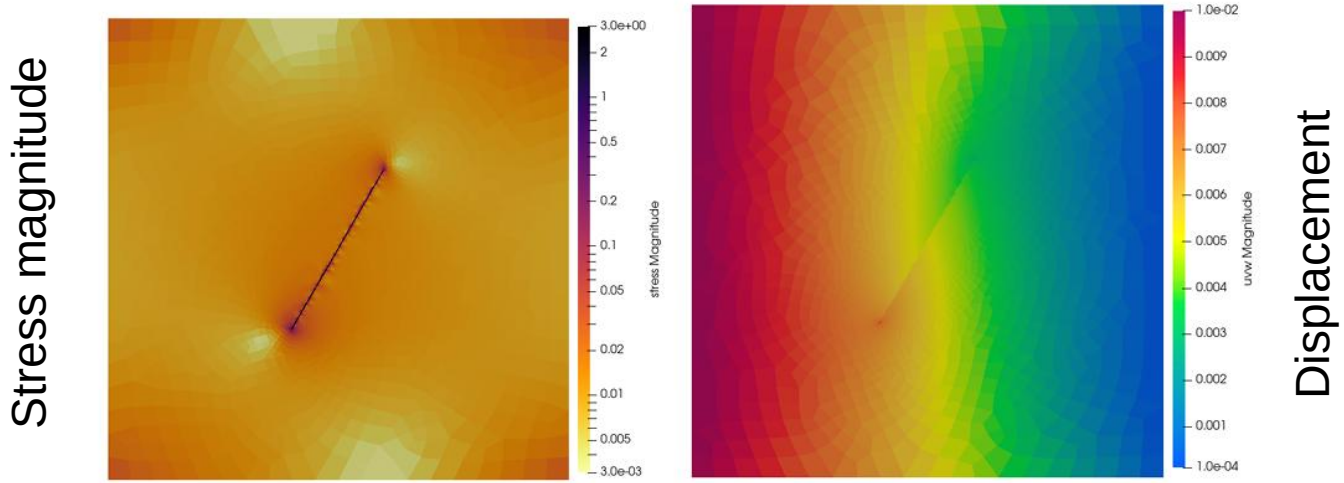


I. Konshin, K. Terekhov / Comp. Math. Math. Phys. (2025)

Multiphysics problems

- Rigid body contact mechanics (4x4 block and 1x1 block at contact)

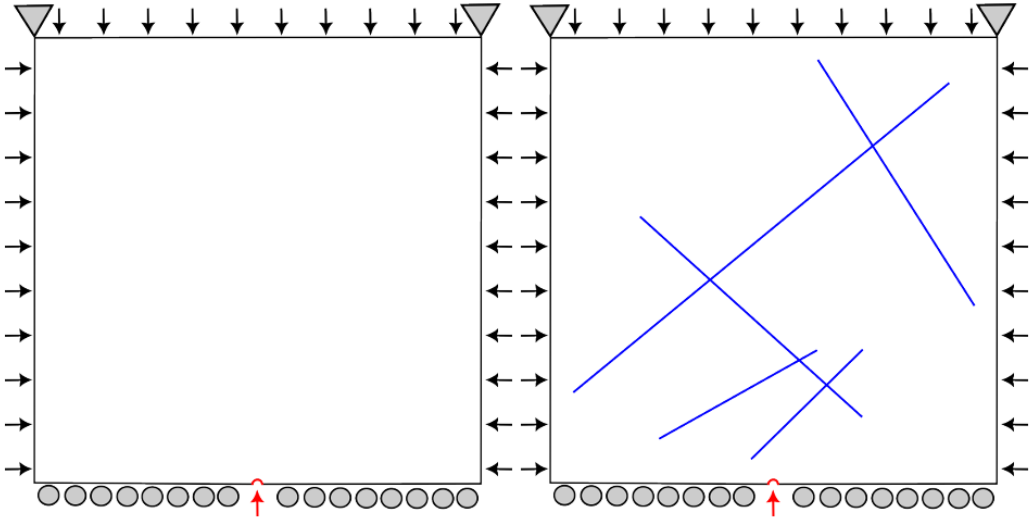


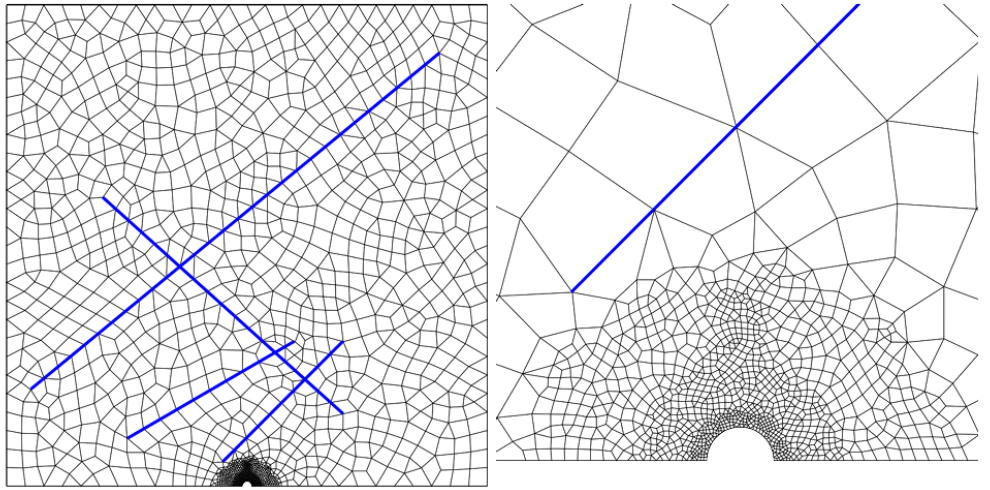


I. Konshin, K. Terekhov / Comp. Math. Math. Phys. (2025)

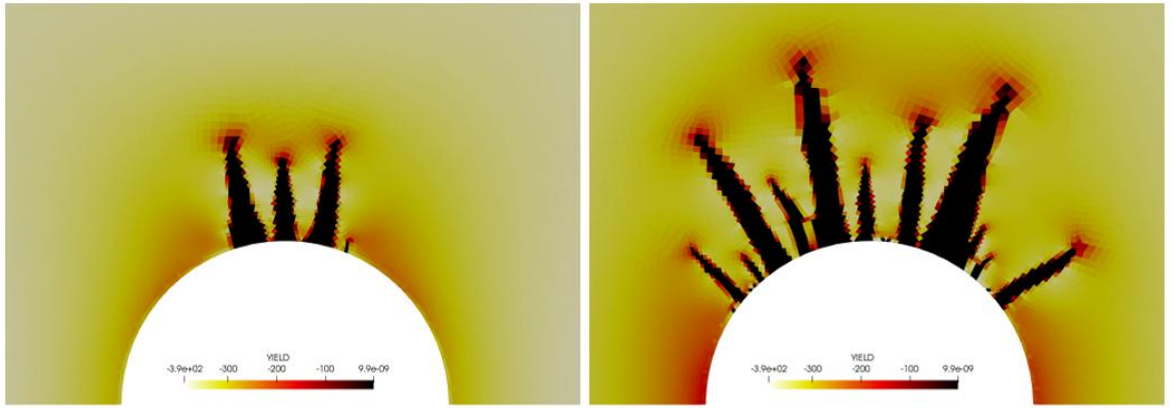
Multiphysics problems

- Two-phase poroelastoplasticity (5x5 block) with fractures (2x2 block)





Yield (no fractures)



Yield (with fractures)

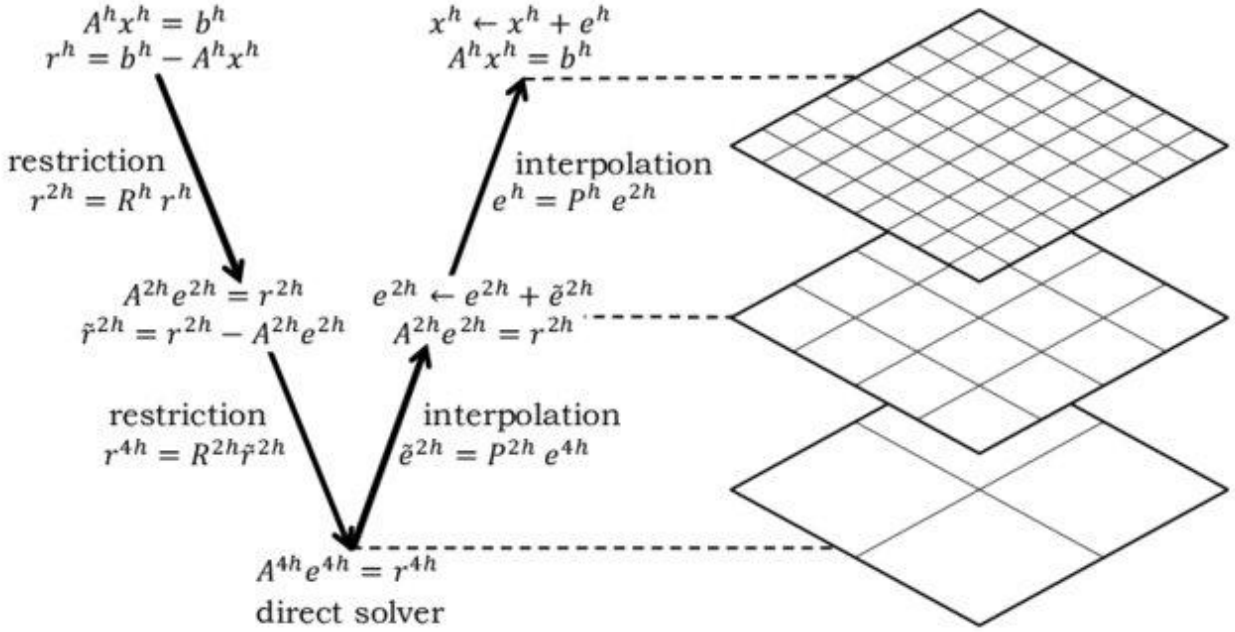
I. Konshin, K. Terekhov / Comp. Math. Math. Phys. (2025)

Competition

- **PETSc**:
  - -ksp_type cg -ksp_relative_tol 1e-7 -ksp_norm_type unpreconditioned
  - -pc_type gamg -pc_gamg_agg_nsmooths 1
- **HYPRE**:
  - -ksp_type cg -ksp_relative_tol 1e-7 -ksp_norm_type unpreconditioned
  - -pc_type hypre -pc_hypre_type boomeramg
  - -pc_hypre_boomeramg_agg_nl 0
  - -pc_hypre_boomeramg_coarsen_type PMIS
  - -pc_hypre_boomeramg_strong_threshold 0.25
  - -pc_hypre_boomeramg_interp_type ext+i
  - -pc_hypre_boomeramg_relax_type_all Chebyshev
- Options are passed directly to HYPRE **bypassing** PETSc

Algebraic multigrid method

- **Setup** phase:
  - Coarse-fine space splitting.
  - Interpolation and restriction operators.
  - Coarse space computation.
  - Smoother or preconditioner setup.
- **Solve** phase:
  - Smoother application.
  - Matrix-vector multiplication.
- **No grids, only matrices!**



(illustration from internet)

References

1) Bakhvalov, Nikolai Sergeevich. ***On the convergence of a relaxation method with natural constraints on the elliptic operator.*** USSR Computational Mathematics and Mathematical Physics 6.5, 101--135 (1966)

2) Fedorenko, Radii Petrovich. ***Iterative Methods for Elliptic Difference Equations.*** Russian Mathematical Surveys 28, 129--195 (1973)

3) Brandt, A and McCormick, S and Ruge, J.: ***Algebraic multigrid (AMG) for automatic algorithm design and problem solution.*** Report,. Comp. Studies, Colorado State University, Ft. Collins (1982)

4) Ruge, John W and Stuben, Klaus.: ***Algebraic multigrid. Multigrid methods***, SIAM, 73--130 (1987)

Our works:

1) I Konshin, K Terekhov: **Adaptive block algebraic multigrid method for multiphysics problems**, Computational Mathematics and Mathematical Physics 65 (7), 1495-1519 (2025)

2) I Konshin, K Terekhov, Y Vassilevski: **Strategies with Algebraic Multigrid Method for Coupled Systems** Lobachevskii Journal of Mathematics 45 (1), 251-261 (2024)

3) I Konshin, K Terekhov: **Block Algebraic Multigrid Method for Saddle-Point Problems of Various Physics** Russian Supercomputing Days, 17-34 (2023)

4) I Konshin, K Terekhov: **Distributed Parallel Bootstrap Adaptive Algebraic Multigrid Method** Russian Supercomputing Days, 92-111 (2022)

5) I Konshin, K Terekhov: **Sparse system solution methods for complex problems** International Conference on Parallel Computing Technologies, 53-73 (2021)

Near Null-Space Vector

- Linear system: $A\mathbf{x} = \mathbf{b}$, where A is $N \times N$ matrix.

- Let $A\mathbf{e} \approx \mathbf{0}$, where vector $\mathbf{e}$ is **near null space** of the system:

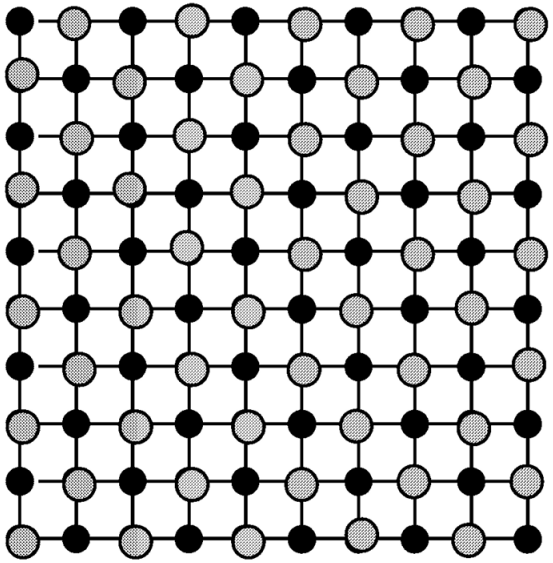
$$a_{ii}\mathbf{e}_i \approx -\sum_{i \neq j} a_{ij}\mathbf{e}_j$$

- For elliptic system the best guess is: $\mathbf{e} = \mathbf{1}$ – **classical** AMG**.**

- **Adaptive multigrid**: exploit information on $\mathbf{e}$ for general systems.
- **Bootstrap process**: try to estimate $\mathbf{e}$ with several iterations of the available smoother
- Ideal $\mathbf{e}$ is an **eigenvector** corresponding to smallest eigenvalue – extremely expensive to find!

Space and Connections Splitting

- **Coarse-fine** splitting of the grid elements: $\Omega = \{1, \dots, N\} = C \cup F$.
- **Connections** of the element: $N_i = \{j \mid i \neq j, a_{ij} \neq 0\}$.
- **Strong-weak** splitting of connections: $N_i = S_i \cup W_i = I_i \cup T_i \cup E_i \cup W_i$.
  - $I_i = S_i \cap C$ – **interpolatory connections**.
  - W_i - weak connections, absorbed by the diagonal coefficient.
  - $T_i \cup E_i = S_i \cap F$ – **strong non-interpolatory** connections.
  - T_i – **twice-removed interpolation**, requires $\forall j \in T_i : S_i \cap S_j \cap C \neq \emptyset$.
  - E_i – **absorbed by the coefficient**, do not satisfy the condition.
- **Ruge-Stuben** rules for the **coarse-fine** splitting (**expensive!**):
  - $\forall i \in F$: $\forall j \in S_i \cap F : S_i \cap S_j \cup C \neq \emptyset$ (E_i is always empty)
  - C is a maximal independent set in the graph of strong connections.



V.E. Henson, U.M. Yang
/ Applied Numerical Mathematics (2002)

Interpolation Method

- Using introduced spaces:

$$a_{ii}\mathbf{e}_i \approx -\sum_{i\neq j} a_{ij}\mathbf{e}_j = -\sum_{j\in I_i} a_{ij}\mathbf{e}_j - \sum_{j\in W_i} a_{ij}\mathbf{e}_j - \sum_{j\in T_i} a_{ik}\mathbf{e}_k - \sum_{j\in E_i} a_{ij}\mathbf{e}_j$$

- Twice-removed interpolation for T_i:

$$a_{ik}\mathbf{e}_k \approx -\sum_{j\in S_i\cap S_k\cap C} \frac{a_{ik}a_{kj}\mathbf{e}_k\mathbf{e}_j}{\sum_{l\in S_i\cap S_k\cap C} a_{kl}\mathbf{e}_l}$$

- Now $A\mathbf{e} \approx \mathbf{0}$ turns into expression:

$$\left(a_{ii} + \sum_{j\in W_i} a_{ij}\frac{\mathbf{e}_j}{\mathbf{e}_i}\right)\mathbf{e}_i \approx -\eta_i \sum_{j\in I_i}\left(a_{ij} + \sum_{k\in T_i} \frac{a_{ik}a_{kj}\mathbf{e}_k}{\sum_{l\in S_i\cap S_k\cap C} a_{kl}\mathbf{e}_l}\right)\mathbf{e}_j$$

- Multiplying coefficient for E_i:

$$\eta_i = \frac{\sum_{k\in S_i} a_{ik}\mathbf{e}_k}{\sum_{k\in S_i\setminus E_i} a_{ik}\mathbf{e}_k}$$

Interpolation Method

- Interpolation:

$$\mathbf{e}_i = \sum_{j \in I_i} \omega_{ij}\, \mathbf{e}_j$$

- Weights:

$$\omega_{ij} = \frac{-\eta_i \mathbf{e}_i}{a_{ii} + \sum_{j \in W_i} a_{ij}\mathbf{e}_j} \left(a_{ij} + \sum_{k \in T_i} \frac{a_{ik} a_{kj} \mathbf{e}_k}{\sum_{l \in S_i \cap S_k \cap C} a_{kl}\mathbf{e}_l} \right)$$

- Prolongator:

$$P_i = \begin{cases} \sum_{j \in I_i} \omega_{ij}\delta_j & i \in F \\ \delta_i & i \in C \end{cases}$$

- Coarse-space system:

$$B = P^T A P$$

Choosing Spaces

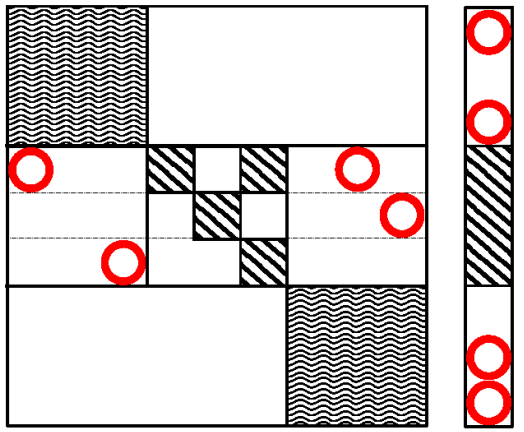
- **Classical** selection of strong connections by Ruge-Stuben:
  - $S_i = \left\{ j \mid -a_{ij} \geq \theta \max\limits_{k \in N_i} (-a_{ik}) \right\}, \quad \theta = \frac{1}{4}.$
- **Modified** selection of strong connections:
  - $S_i = \left\{ j \mid -\operatorname{sgn}(a_{ii}\mathbf{e}_i)\, a_{ij}\mathbf{e}_j \geq \theta \max\limits_{k \in N_i} (-\operatorname{sgn}(a_{ii}\mathbf{e}_i) a_{ik}\mathbf{e}_k) \right\}, \quad \theta = \frac{1}{4}$
  - Additional requirement: $a_{ii}\mathbf{e}_i \left(a_{ii}\mathbf{e}_i + \sum_{j \in W_i} a_{ij}\mathbf{e}_j \right) > 0.$
  - **Important** on coarser levels.
- **Faster** operator construction without **twice-removed interpolation**, all coefficients are absorbed by the **diagonal coefficient**.

Necessary steps

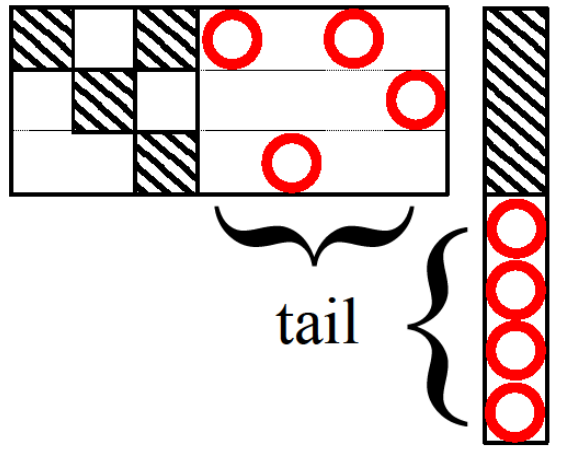
- **Setup**:
  - Compute operator:
    - maximal **weighted** independent set,
    - compute stencil and weights,
    - assemble tall sparse matrix P.
  - Compute matrix **transpose** for $R = P^T$.
  - Compute matrix **triple product** $B = R * A * P$.
- **Iterations**:
  - Matrix-vector product.
  - Smoothers: Chebyshev or Gauss-Seidel.

Distributed version

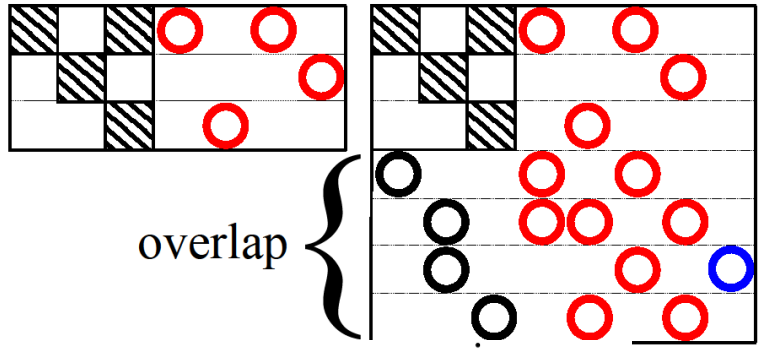
- **Setup**:
  - Compute operator:
    - Matrix overlapping
    - Luby's algorithm for maximal independent set,
  - Compress to super-sparse format for matrix **transpose**
  - Parallel Gustavson's sparse matrix-matrix **product**
  - Communicator reduction with sparse matrix redistribution
- **Iterations**:
  - Parallel matrix-vector product
  - Multicoloring for Gauss-Seidel



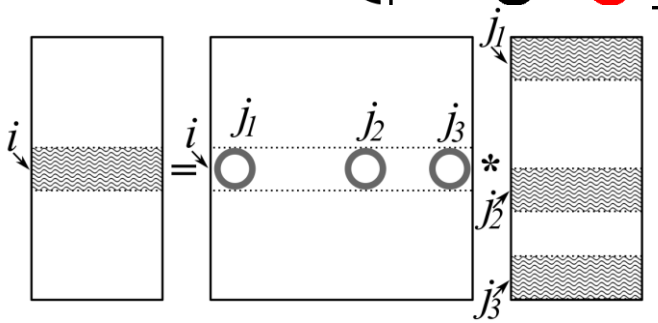
Vector extension



Matrix-vector local ordering

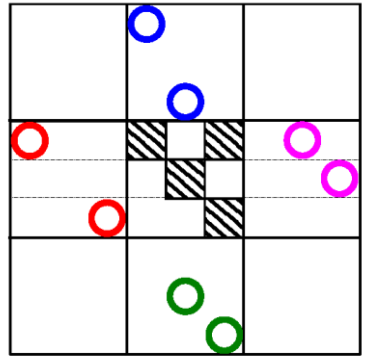


Matrix overlapping algorithm



Gustavson's algorithm

Distributed matrix transpose



Competition overview

- **PETSc**:
  - Requires computing **square graph** $G^T G$ for aggressive coarsening
  - Requires finding **eigenvector** for smoothed aggregation
  - Longer setup but very low iteration count
- **HYPRE**:
  - **Distance-two** interpolation 

  - **Prolongation truncation** to reduce operator complexity
  - Shorter setup, larger iteration count
- Both estimate **largest eigenvalue** for Chebyshev smoother:
  - PETSc automatically with eigenvector, HYPRE performs 4 CG iterations