Greedy Dissection Method for Shared Parallelism in Incomplete Factorization within INMOST Platform

Kirill Terekhov[1]

[1]Marchuk Institute of Numerical Mathematics of the Russian Academy of Sciences

Russian Supercomputing Days

Septermber 28, 2021

INMOST

INMOST (www.inmost.org, www.inmost.ru) is a short for:

Integrated

Numerical

Modeling and

Object-oriented

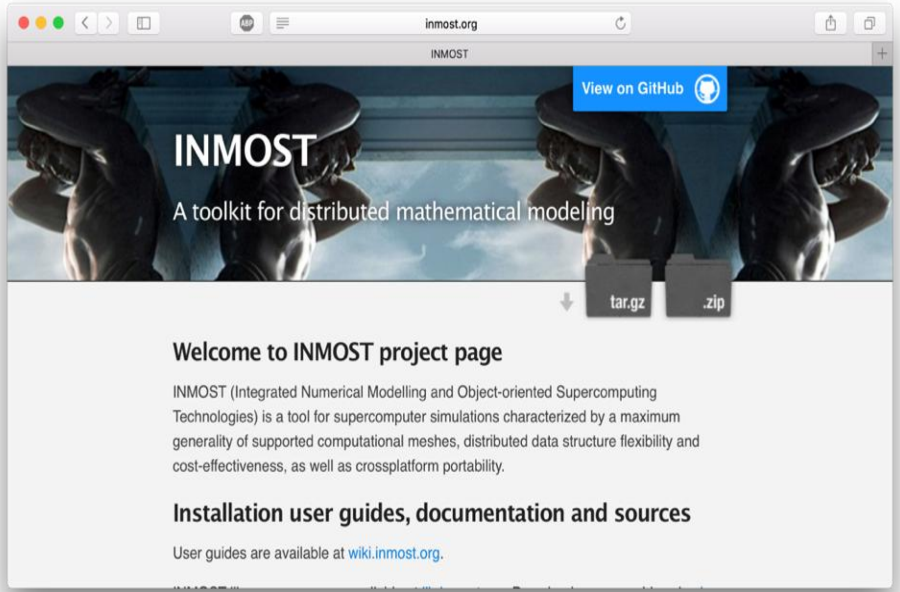
Supercomputing

Technologies

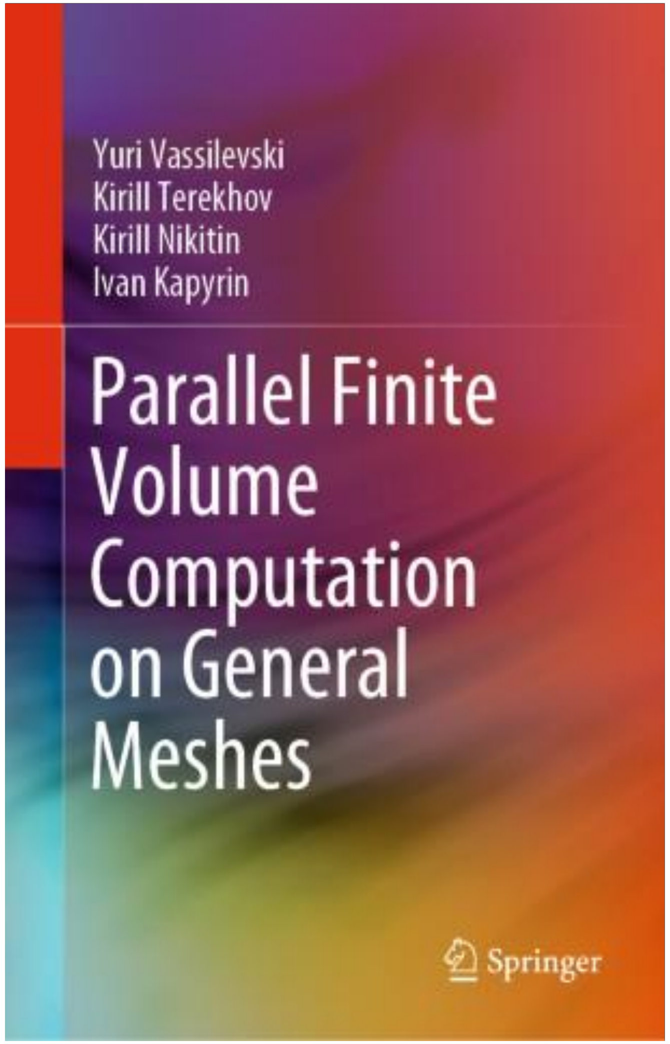
- Distributed meshes
- Distributed linear system assembly
- Parallel linear solver
- Automatic differentiation
- Nonlinear system assembly
- Coupling of unknowns and models

First version during 2012 internship at ExxonMobil

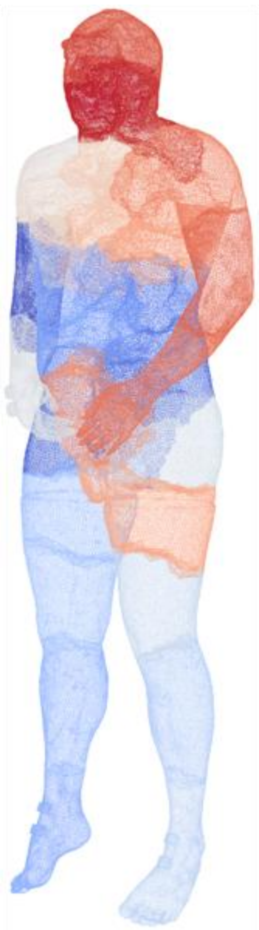
Contributors: Igor Konshin, Kirill Nikitin, Alexander Danilov, Ivan Kapyrin, Yuri Vassilevski, Alexei Chernyshenko (INM RAS, IBRAE RAS), Igor Kaporin (CMC RAS) Dmitri Bagaev, Andrei Burachkovski (MSU), Ruslan Yanbarisov, Alexei Logkiy, Sergei Petrov, Ivan Butakov (MIPT), German Kopytov (BFU), Timur Garipov, Pavel Tomin, Christine Mayer (Stanford), Ahmad Abushaikha, Longlong Li (HBKU), et al


inmost.org
INMOST
View on GitHub
INMOST
A toolkit for distributed mathematical modeling
tar.gz
.zip
Welcome to INMOST project page
INMOST (Integrated Numerical Modelling and Object-oriented Supercomputing Technologies) is a tool for supercomputer simulations characterized by a maximum generality of supported computational meshes, distributed data structure flexibility and cost-effectiveness, as well as crossplatform portability.
Installation user guides, documentation and sources
User guides are available at wiki.inmost.org.

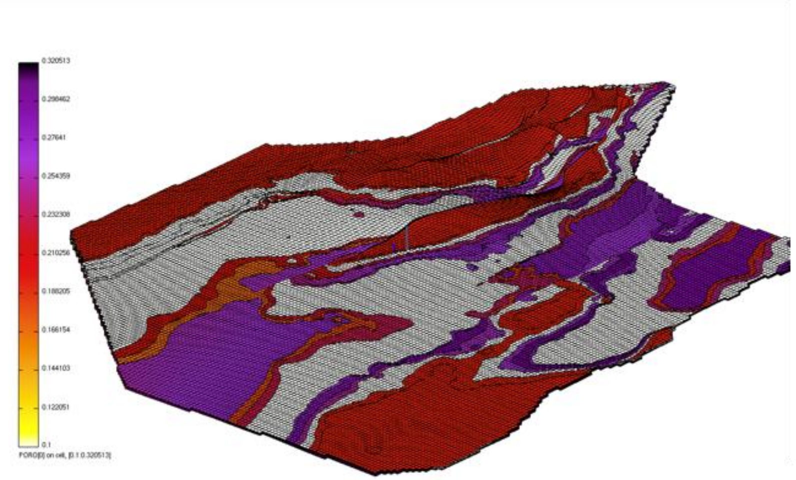

Серия
Суперкомпьютерное
Образование
Ю. В. Василевский, И. Н. Коньшин,
Г. В. Копытов, К. М. Терехов
INMOST ПРОГРАММНАЯ ПЛАТФОРМА И ГРАФИЧЕСКАЯ СРЕДА
ДЛЯ РАЗРАБОТКИ ПАРАЛЛЕЛЬНЫХ ЧИСЛЕННЫХ МОДЕЛЕЙ
НА СЕТКАХ ОБЩЕГО ВИДА
INMOST
ПРОГРАММНАЯ ПЛАТФОРМА
И ГРАФИЧЕСКАЯ СРЕДА

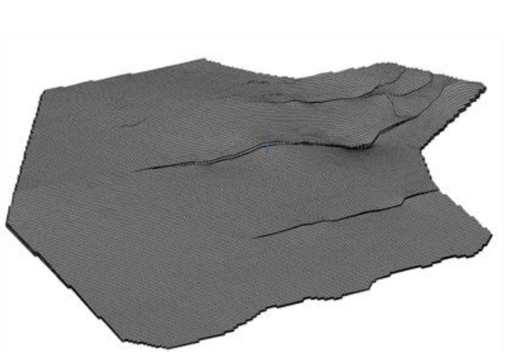

Yuri Vassilevski
Kirill Terekhov
Kirill Nikitin
Ivan Kapyrin
Parallel Finite Volume Computation on General Meshes
Springer

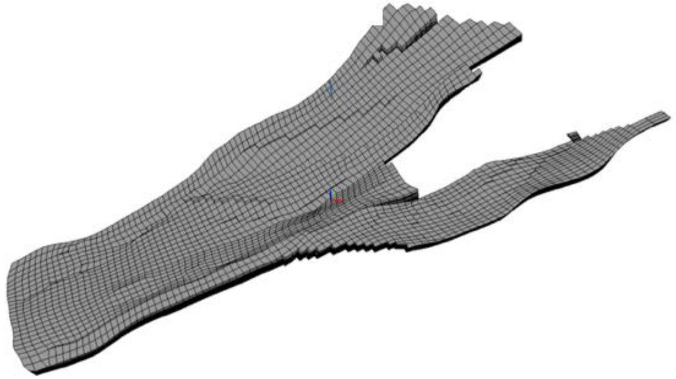
Grids



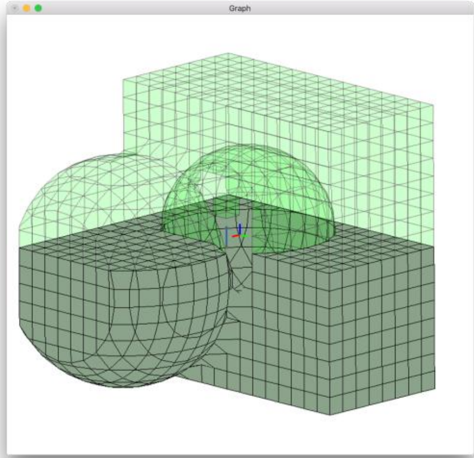
Human body



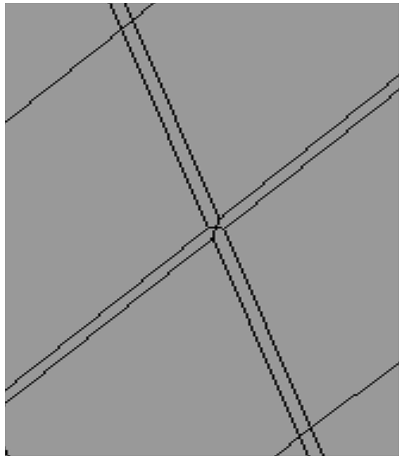




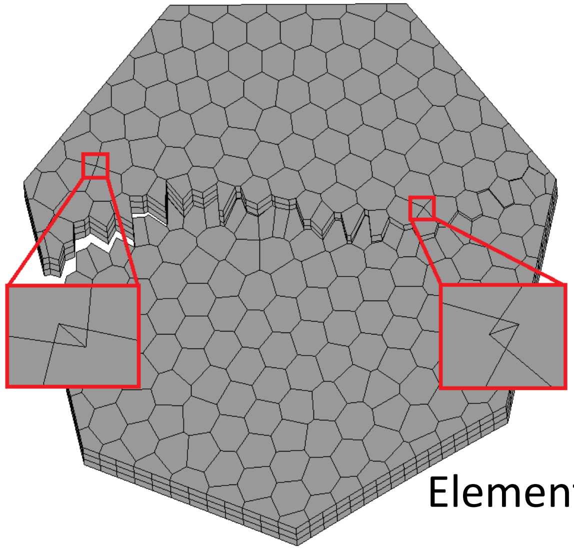
Geological grids with faults and pinch-outs, support for commercial formats of oil & gas simulators



Complex modifications

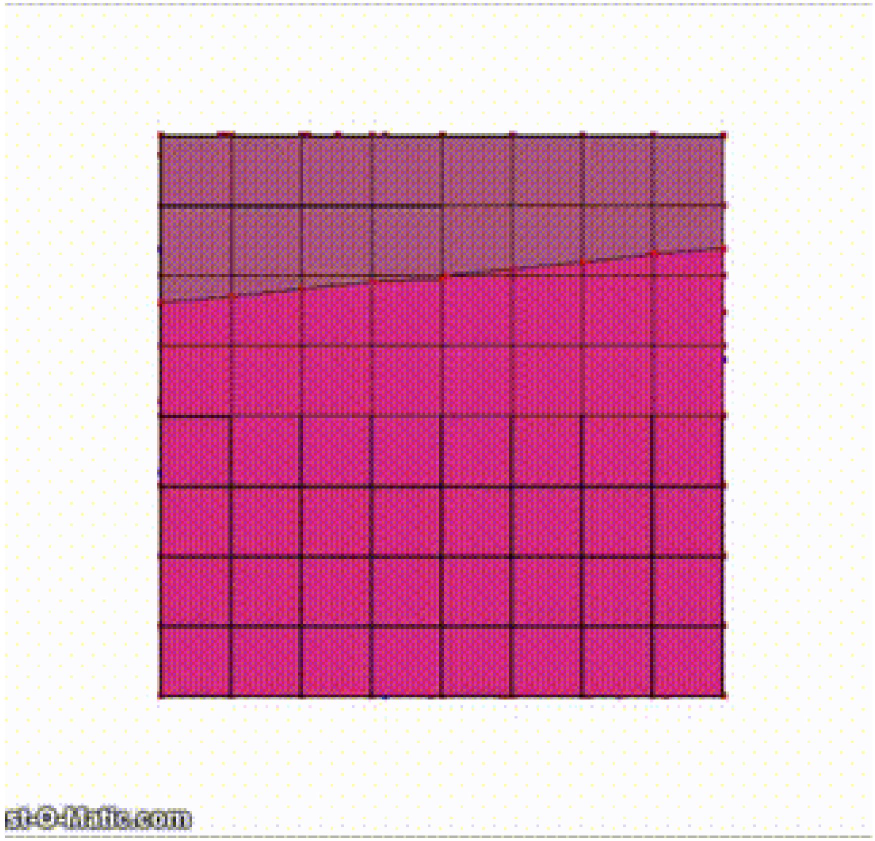


Fracture opening

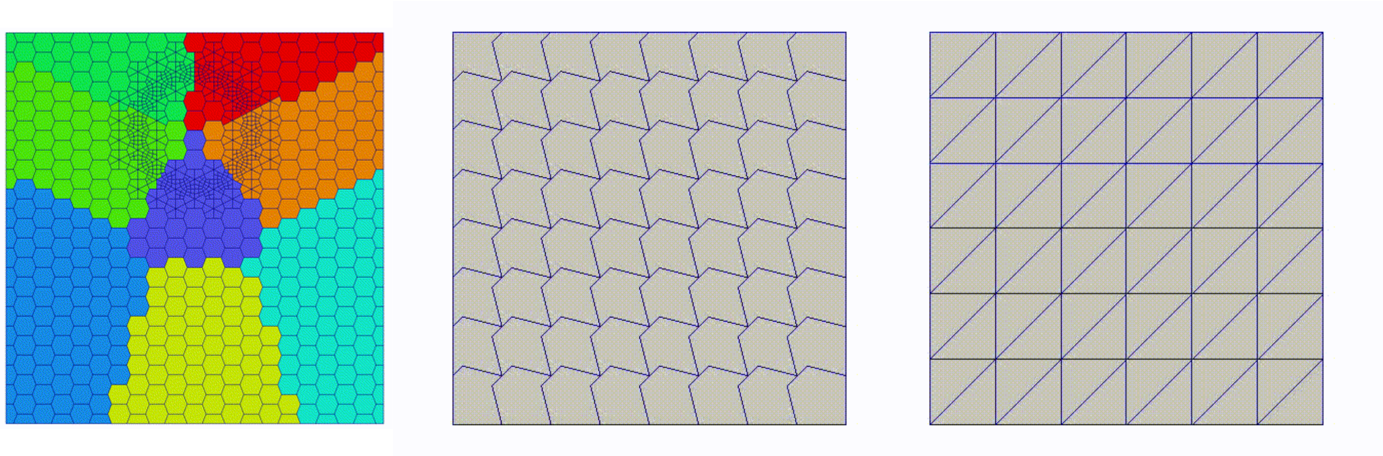


Element collapse

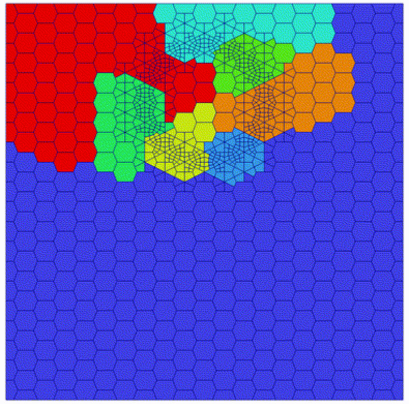
Dynamic grids

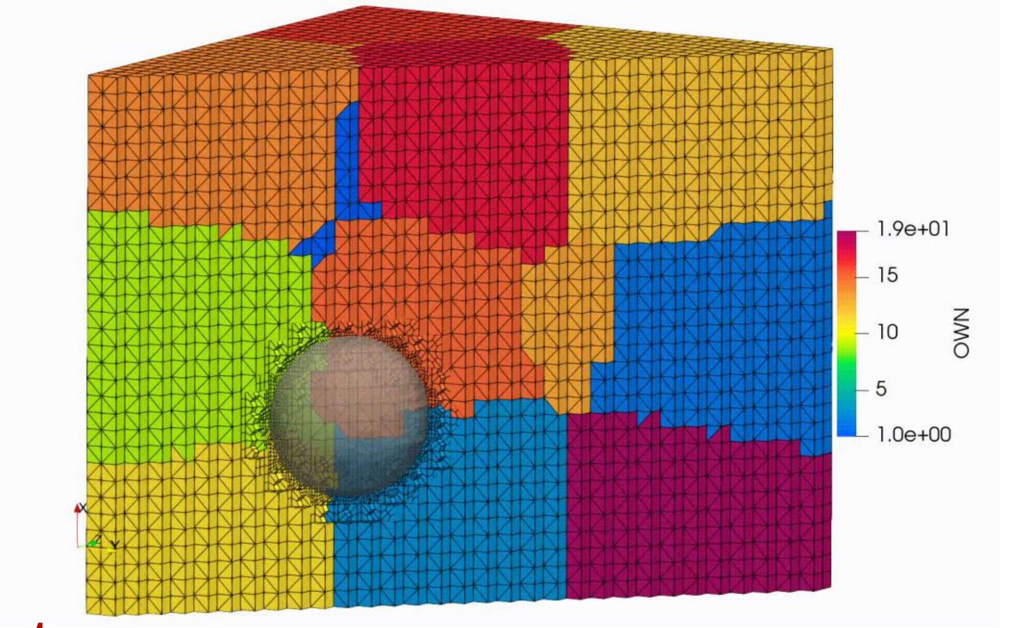


OctreeCutcell example in INMOST-Graphics repository



AdaptiveMesh example for general grid adaptation





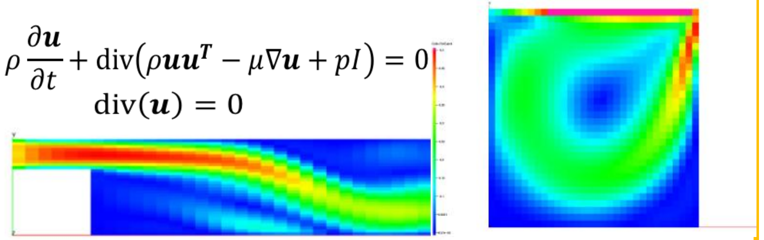
Parmetis_AdaptiveRepart

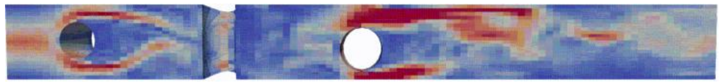
Models

Navier-Stokes

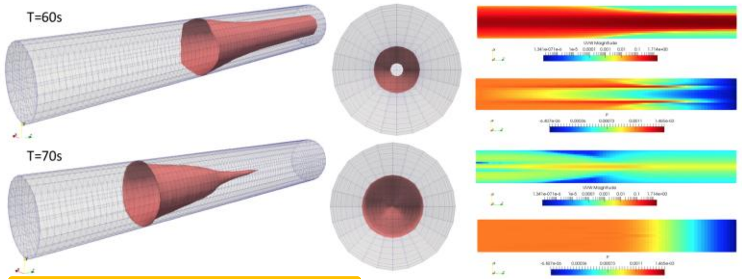
$$\rho \frac{\partial \boldsymbol{u}}{\partial t} + \mathrm{div}\left(\rho \boldsymbol{u}\boldsymbol{u}^T - \mu \nabla \boldsymbol{u} + pI\right) = 0$$

$$\mathrm{div}(\boldsymbol{u}) = 0$$





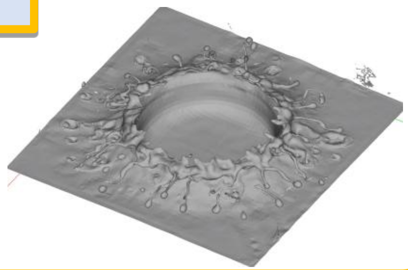
Blood coagulation



Freesurface flows

$$\frac{\partial \varphi}{\partial t} + \mathrm{div}(\varphi \boldsymbol{u}) = 0$$

$$|\nabla \varphi| = 1$$

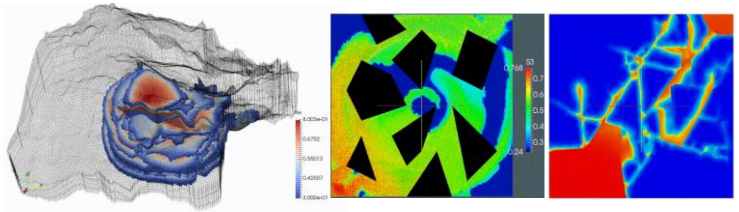


Multiphase filtration

$$\frac{\partial \rho_w \theta S_w}{\partial t} - \nabla \cdot \left(\lambda_w \mathbb{K}(\nabla p - \rho_w g \nabla z)\right) = q_w$$

$$\frac{\partial \rho_o \theta S_o}{\partial t} - \nabla \cdot \left(\lambda_o \mathbb{K}(\nabla p - \nabla P c_o - \rho_w g \nabla z)\right) = q_o$$

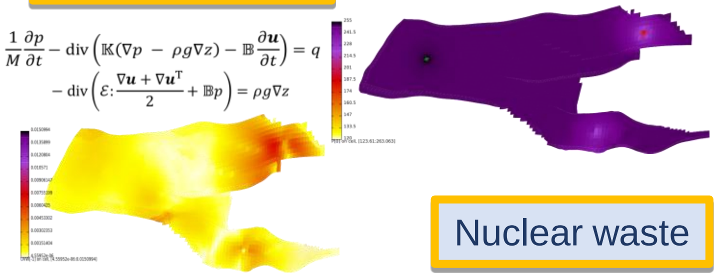
$$\frac{\partial \rho_g \theta (R S_o + S_g)}{\partial t} - \nabla \cdot \left(\lambda_g \mathbb{K}(\nabla p - \nabla P c_g - \rho_g g \nabla z)\right) = q_g$$



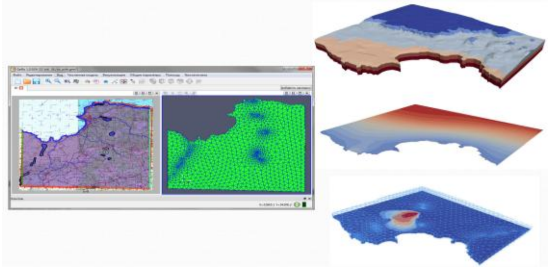
Poromechanics

$$\frac{1}{M}\frac{\partial p}{\partial t} - \mathrm{div}\left(\mathbb{K}(\nabla p - \rho g \nabla z) - \mathbb{B}\frac{\partial \boldsymbol{u}}{\partial t}\right) = q$$

$$-\mathrm{div}\left(\mathcal{E}\colon \frac{\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T}{2} + \mathbb{B}p\right) = \rho g \nabla z$$

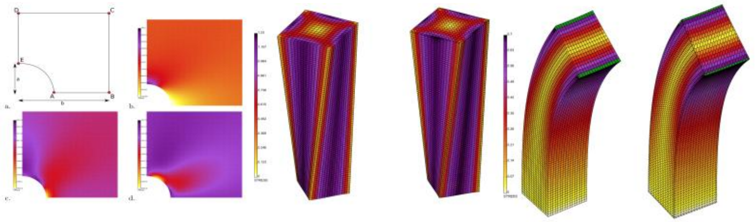


Nuclear waste

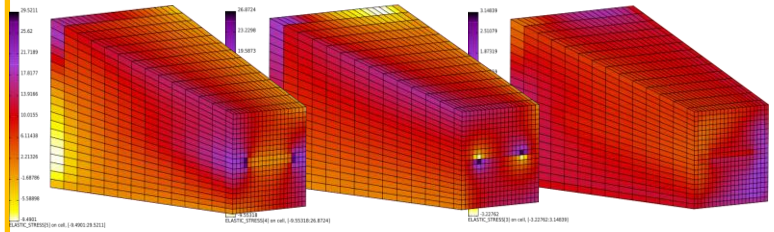


Mechanics

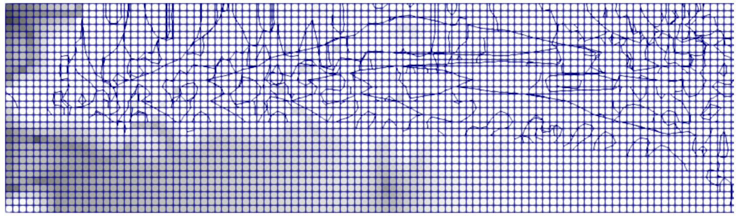
$$-\mathrm{div}(\boldsymbol{\sigma}) = 0, \qquad \mathrm{C}\colon \boldsymbol{\sigma} = \frac{\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T}{2}$$



Contact mechanics



Fracturing



Jacobian Structure in Multiphysics Problem

Models connect through

- **functions**: mobility, density, porosity, properties,…
- **fluxes**: Biot term, incompressibility conditions, capillary pressure, …
- **right hand side:** reactions, …

Ability to **control**:

- models, unknowns, functions, coupling conditions, residual and Jacobian assembly.

Fully implicit solution:

- saddle system, inf-sup stability issue and **complex linear systems**

Model decomposition for **reservoir simulator:**

Equations \ Unknowns	u,v,w	p	S_o, S_w, S_g	T
Elasticity	Deformation	Biot term	Capillary effect	Thermal expantion
Darcy law	Biot term, porosity	Flow	Capillary effect	Thermal conduction
Transport	Structure movement	Flow velocity, mobility	Concentrations	Heat gradient
Heat conduction	Structure movement	Flow velocity	Density gradient	Thermal

Large systems

- Navier-Stokes system:

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \operatorname{div}(\rho \boldsymbol{u}\boldsymbol{u}^T - \mu \nabla \boldsymbol{u} + p\mathbb{I}) = -\frac{\mu}{K_f}\boldsymbol{u},$$

$$\operatorname{div}(\rho \boldsymbol{u}) = 0,$$

- Prothrombin (II):

$$\frac{\partial P}{\partial t} + \operatorname{div}(P\boldsymbol{u} - D\nabla P) = -(k_1\phi_c + k_2 B_a + k_3 T + k_4 T^2 + k_5 T^3)P,$$

- Thrombin (IIa):

$$\frac{\partial T}{\partial t} + \operatorname{div}(T\boldsymbol{u} - D\nabla T) = (k_1\phi_c + k_2 B_a + k_3 T + k_4 T^2 + k_5 T^3)P - k_6 AT,$$

- Clot formation factors (IXa, Xa):

$$\frac{\partial B_a}{\partial t} + \operatorname{div}(B_a\boldsymbol{u} - D\nabla B_a) = (k_7\phi_c + k_8 T)(B_0 - B_a) - k_9 AB_a,$$

- Antithrombin (ATIII):

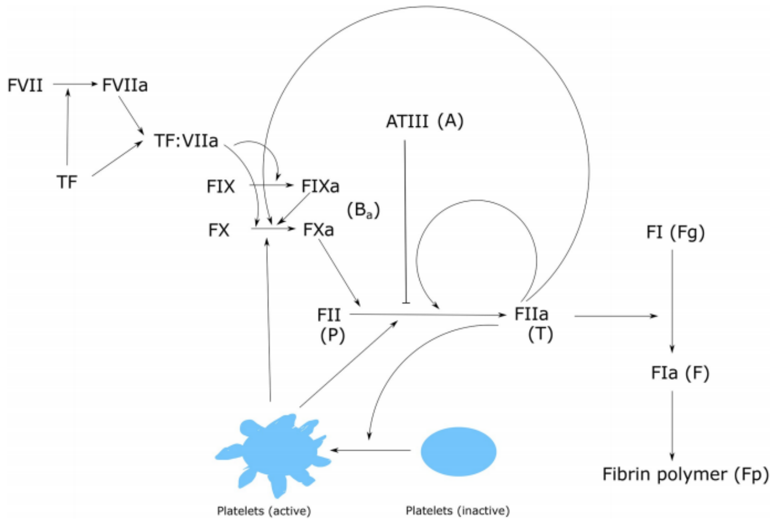
$$\frac{\partial A}{\partial t} + \operatorname{div}(A\boldsymbol{u} - D\nabla A) = -k_6 A - k_9 AB_a,$$

- Fibrinogen (I):

$$\frac{\partial F_g}{\partial t} + \operatorname{div}(F_g\boldsymbol{u} - D\nabla F_g) = -\frac{k_{10} T F_g}{K_{10} + F_g},$$

- Fibrin (Ia):

$$\frac{\partial F}{\partial t} + \operatorname{div}(F\boldsymbol{u} - D\nabla F) = \frac{k_{10} T F_g}{K_{10} + F_g} - k_{11} F,$$



- Fibrin-polymer:

$$\frac{\partial F_p}{\partial t} = k_{11} F,$$

- Inactivated platelets:

$$\frac{\partial \phi_f}{\partial t} + \operatorname{div}\Big(k(\phi_c, \phi_f)(\phi_f \boldsymbol{u} - D_p \nabla \phi_f)\Big) = (k_{12} T - k_{13}\phi_c)\phi_f,$$

- Activated platelets:

$$\frac{\partial \phi_c}{\partial t} + \operatorname{div}\Big(k(\phi_c, \phi_f)(\phi_c \boldsymbol{u} - D_p \nabla \phi_c)\Big) = -(k_{12} T - k_{13}\phi_c)\phi_f,$$

- Platelets mobility:

$$k(\phi_c, \phi_f) = \tanh\left(\pi\left(1 - \frac{\phi_c + \phi_f}{\phi_{max}}\right)\right),$$

- Permeability :

$$\frac{1}{K_f} = \frac{16}{\alpha^2}\phi_p^{\frac{3}{2}}(1 + 56\phi_p)\frac{\phi_{max} + \phi_c}{\phi_{max} - \phi_c}, \qquad \phi_p = \min\left(\frac{7}{10}, \frac{F_p}{7000}\right).$$

Bouchnita, A., Terekhov, K., Nony, P., Vassilevski, Y., & Volpert, V.: ***A mathematical model to quantify the effects of platelet count, shear rate, and injury size on the initiation of blood coagulation under venous flow conditions*. PloS one,** 15(7), e0235392, 2020

Large problems

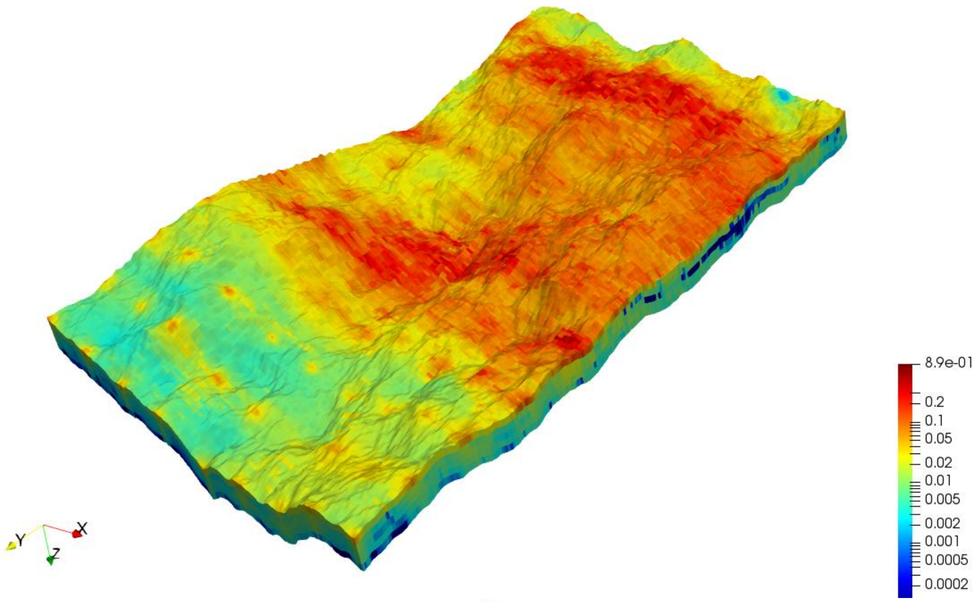
- Suitable for large problem solutions:
  - Black oil problem (previous talk)
  - 3x unknowns per cell
  - **100M** and **200M** cells (320 cores, INM RAS cluster):

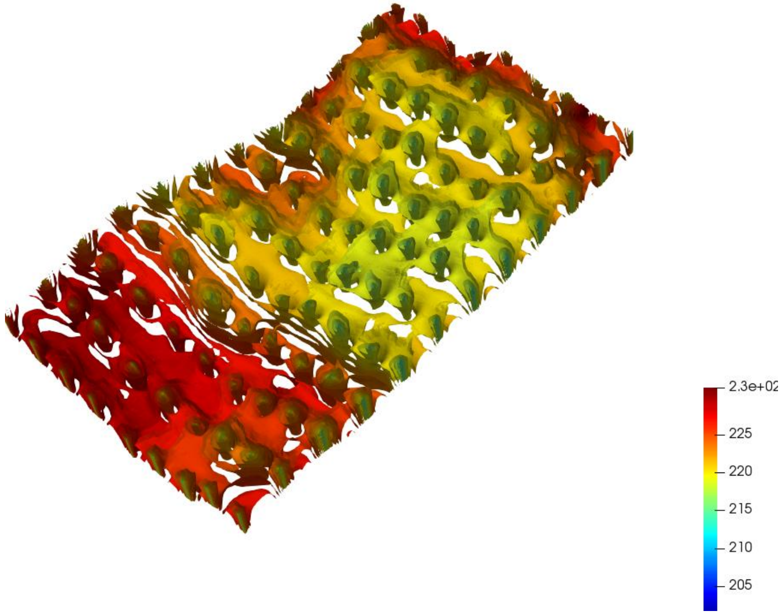
Case	T_{mat}	T_{prec}	T_{iter}	T_{sol}	T_{upd}	N_n	N_l
SPE10_100M	14	18.5	55.4	78.6	0.2	402	3.5
SPE10_200M	29.6	34.7	64.1	107.5	0.38	428	3.96

  - **Memory** per core (**significant issue!**):

Case	M_{grid}	M_{mat}	M_{prec}	M_{tot}
SPE10_100M	856.7	165.6	558.4	1943.6
SPE10_200M	1624	346.5	1054.3	4365.6

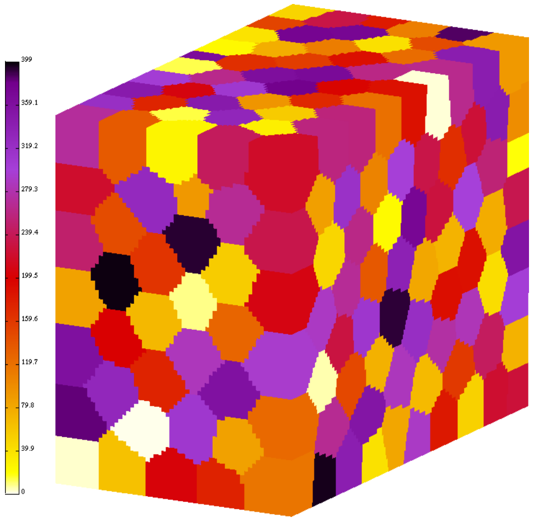
- Scaled up to **1B of cells** on 9600 Cray cores by Ahmad Abushaika and Longlong Li at HBKU, Qatar.

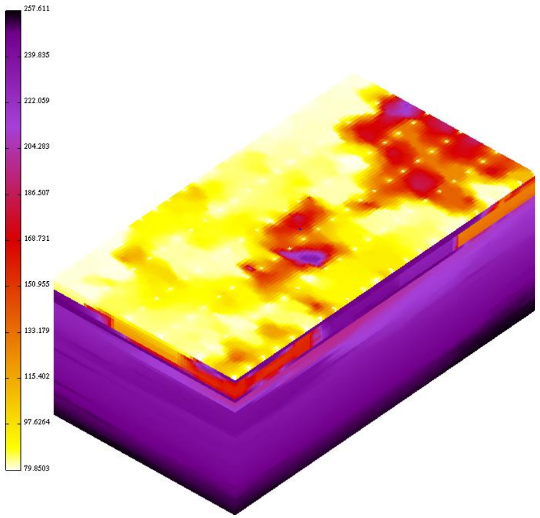


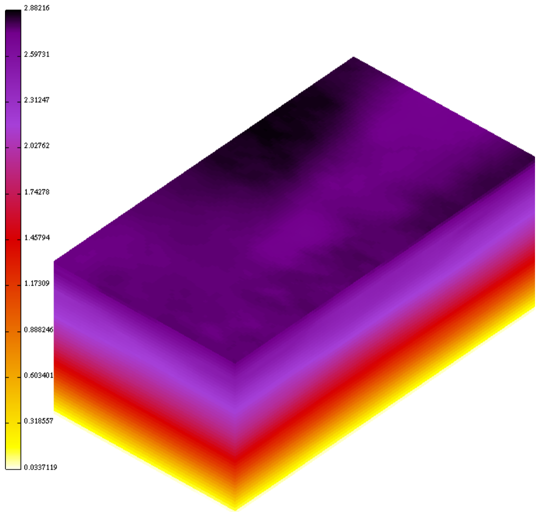


Large problems

- Suitable for large problem solutions:
  - Poroelasticity problem
  - 4x unknowns per cell
  - **1.2M** cells (INM RAS cluster, Lomonosov):







Machine	N_{proc}	T_{tot}	T_{asm}	T_{prec}	T_{iter}	T_{upd}
INM RAS cluster	100	15079.4	1119.8	7245.2	4463	479.7
	200	8791.2	582.9	3926.2	2800.9	252.4
	400	4637	300.3	1965.6	1374.2	127
Lomonosov supercomputer	700	3536	234.1	1071.1	1112.42	70.5

  - **Target**: full-field geomechanical well stability test.

Flagship INMOST Linear Solver

- Preconditioned **BiCGStab(l)** method[1].
- Preconditioner **MPI-parallelization** using **Additive Schwarz Method**.
- Preconditioner **OpenMP-parallelization** using **Bordered Block-Diagonal Form**[2,3]. **(New!)**
- **Multi-level preconditioner** with **deferred pivoting** based on the second-order **Crout-ILU**[4,5].
- **Condition estimation** of the inverse factors determines the **coarse system** and tunes dropping tolerances[6,7].
- **Scaling** and **reordering** of the local system before **factorization**[8,9,10].

References

1) Sleijpen, G.L.G., Diederik R. F.**:** ***BiCGstab (l) for linear equations involving unsymmetric matrices with complex spectrum.*** **Electronic Transactions on Numerical Analysis** 1.11 (1993): 2000**.** **(Krylov method)**

2) Grigori, L., Boman, E. G., Donfack, S., Davis, T. A: ***Hypergraph-based unsymmetric nested dissection ordering for sparse LU factorization***. **SIAM Journal on Scientific Computing**, *32*.6 (2010): 3426-3446. **(Bordered block-diagonal form)**

3) Duff, I. S., Scott, J. A.: ***Stabilized bordered block diagonal forms for parallel sparse solvers*****. Parallel Computing**, 31.3-4 (2005): 275-289. **(Bordered block-diagonal form)**

4) Li N., Saad Y., Chow E.: ***Crout versions of ILU for general sparse matrices.*** **SIAM Journal on Scientific Computing** 25.2 (2003): 716-728. **(Crout-ILU)**

5) Kaporin, I.E.: ***High quality preconditioning of a general symmetric positive definite matrix based on its UTU+ UTR+ RTU-decomposition*****. Numerical linear algebra with applications** 5.6 (1998): 483-509. **(Second-order ILU)**

6) Bollhöfer, M.: ***A robust ILU with pivoting based on monitoring the growth of the inverse factors*****. Linear Algebra and its Applications** 338.1-3 (2001): 201-218. **(Tuning dropping tolerances)**

7) Bollhöfer, M., Saad Y.: ***Multilevel preconditioners constructed from inverse-based ILUs*****. SIAM Journal on Scientific Computing** 27.5 (2006): 1627-1650. **(Computing coarse system)**

8) Cuthill, E., McKee J.: ***Reducing the bandwidth of sparse symmetric matrices.*** **Proceedings of the 1969 24th national conference.** 1969. **(Reordering for bandwidth reduction)**

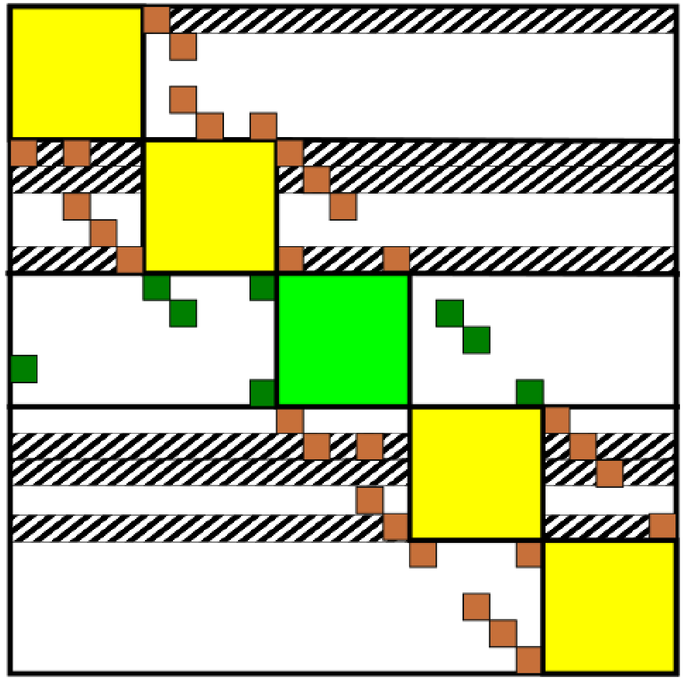
9) Olschowka, M., Arnold N.: ***A new pivoting strategy for Gaussian elimination*****. Linear Algebra and its Applications** 240 (1996): 131-151. **(Maximizing diagonal product)**

10) Kaporin, I.E.: ***Scaling, reordering, and diagonal pivoting in ILU preconditionings*****. Russian Journal of Numerical Analysis and Mathematical Modelling** 22.4 (2007): 341-375. **(Rescaling for condition reduction)**

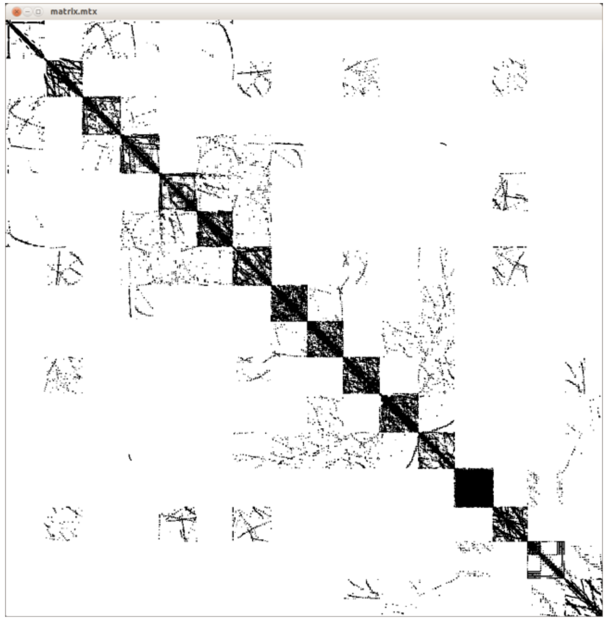
Additive Schwarz Method

- For **MPI-parallelization**.
- Global matrix is composed of local **blocks**.
- Extend blocks to localize the **connection**.
- **Restricted** version.

Matrix overlapping



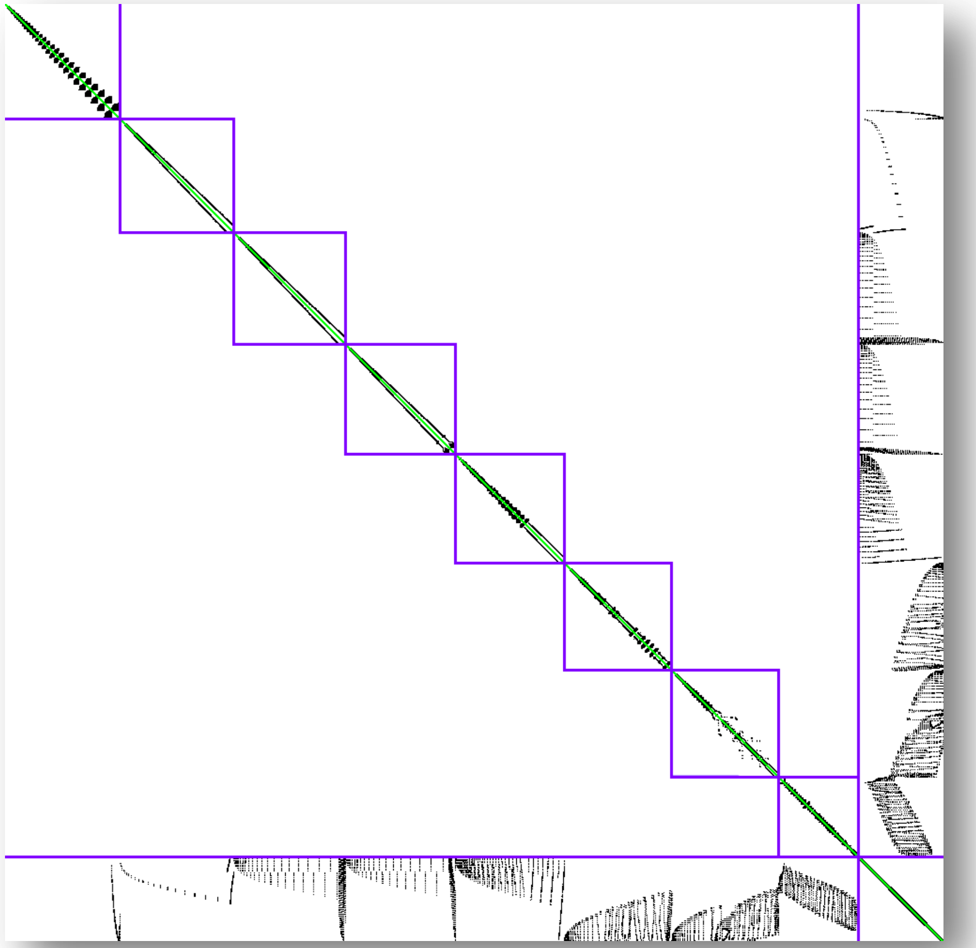
- Local partition outlier
- Remote partition outlier
- Local partition
- Remote partitions
- Extended rows



Distributed system

Bordered Block-Diagonal Form

- For **OpenMP-parallelization**.
- The problem is to find **minimum separator** for the **K**-way partitioning. (**NP-hard**)
- All algorithms are approximate:
  - Greedy dissection (**this work**).
  - Usual graph packages: Metis, Patoh, Mondriaan. (**todo**)
- Work with each block in **parallel**.
- **Differed pivoting** for the **separator**.



Bordered Block-Diagonal Form

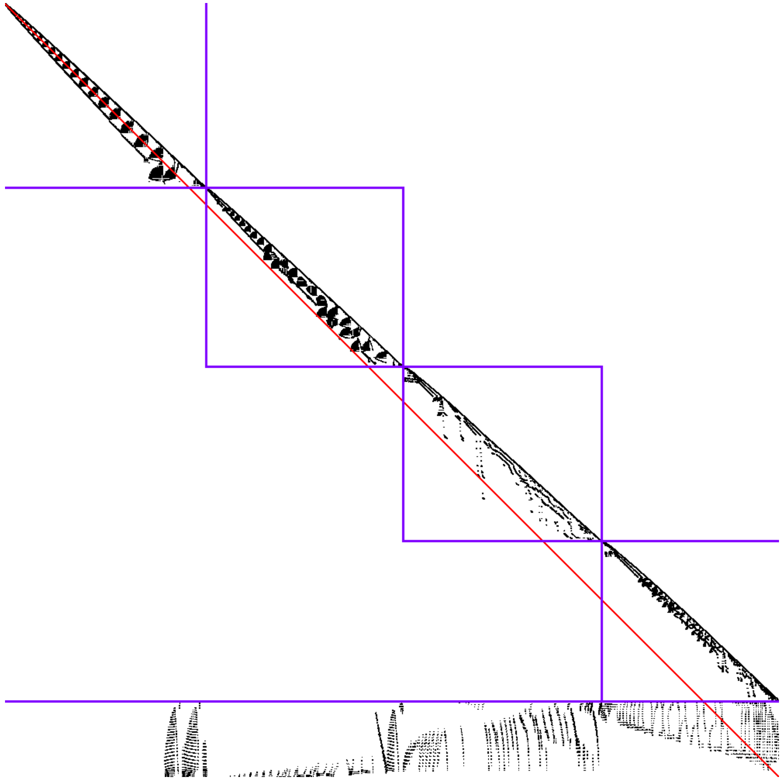
- Greedy algorithm:
  - G – graph of A, H = G^TG – graph product.
  - $|G_i|$ predicts **block size** growth.
  - $|H_i|$ predicts **separator** growth.
  - Using **priority queue** pick a node with minimal growth in both **block** and **separator**.
  - Remove the node from G and H graphs and update **priority queue**.
  - **Sequential ☹**

Algorithm 1 Greedy dissection.

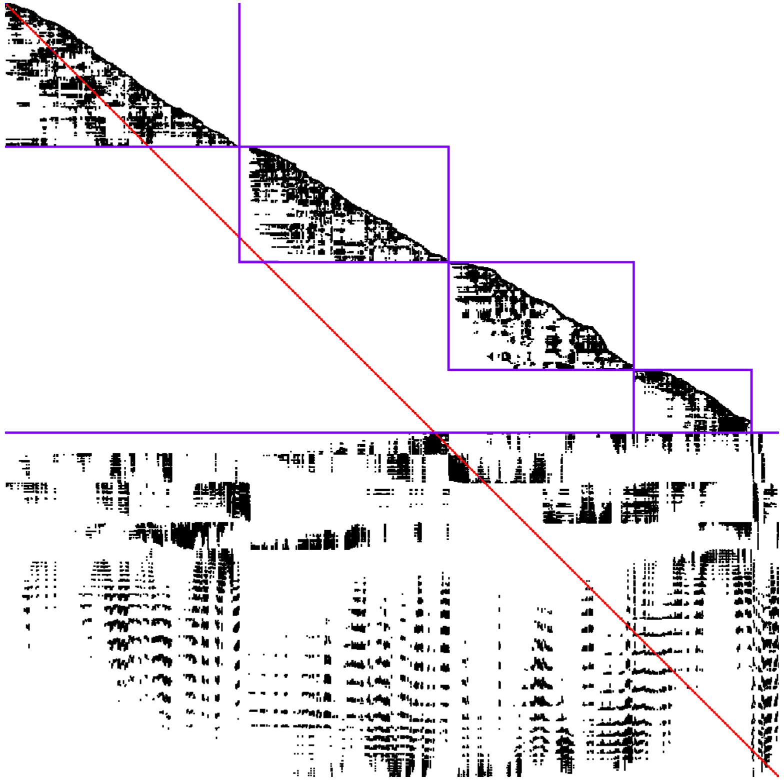
```
function GreedyDissection(A, parts)
    Let G = (V, E), where                        ▷ Graph induced by matrix
    V = {1, ..., n},                             ▷ Vertex set
    G_i = {δ_ij | a_ij ≠ 0}.                     ▷ Edge set
    Let H = G^T G.                               ▷ Graph product
    Let ω_B = {ω_B^i = ||G_i|| | ∀i ∈ V}.        ▷ Weight for block growth
    Let ω_S = {ω_H^i = ||S_i|| | ∀i ∈ V}.        ▷ Weight for separator growth
    Let B_i = {∅}, ∀i ∈ {1, ..., parts}.         ▷ Blocks
    Let S = ∅.                                   ▷ Separator
    Let R = C = V.                               ▷ Candidate row and column sets
    Let P_i = Q_i = 0, ∀i ∈ {1, ..., n}.         ▷ Rows and columns reordering matrices
    r = c = q = 1.                               ▷ Row and column index and current block
    while R ≠ ∅ do
        i = arg min_{j∈R} (ω_B^j + ω_H^j),       ▷ Minimal growth of both the block and separator
        R = R \ {i},                             ▷ Remove from the candidate set
        P_i = r,
        r = r + 1,                               ▷ Enumerate row
        B_q = B_q ∪ {i},                         ▷ Augment current block
        if i ∈ S then S = S \ {i}.
        for m ∈ H_i ∩ R do
            ω_m^S = ω_m^S − 1.                   ▷ Update weights for separator growth
        end for
        for j ∈ G_i ∩ C do
            C = C \ {j},
            Q_j = c,
            c = c + 1.                           ▷ Enumerate column
            for k ∈ G_j^T do
                ω_k^B = ω_k^B − 1.               ▷ Update weights for block growth
                if k ∉ S then
                    S = S ∪ {k},
                    ω_k^S = ω_k^S − 1.           ▷ Add row to separator
                    for m ∈ H_k ∩ R do
                        ω_m^S = ω_m^S − 1.       ▷ Update weights for separator growth
                    end for
                end if
            end for
        end for
        if ||B_q|| ≥ ||R||/parts then
            R = R \ S,                           ▷ Remove current separator from candidate set
            q = q + 1.                           ▷ Consider next block
        end if
    end while
    for all i ∈ S do
        P_i = r,
        r = r + 1.
    end for
    Q = P.                                       ▷ Match row and column indices.
    return [P,Q,B,S]
end function
```

Singly-Bordered Block-Diagonal Form

First level

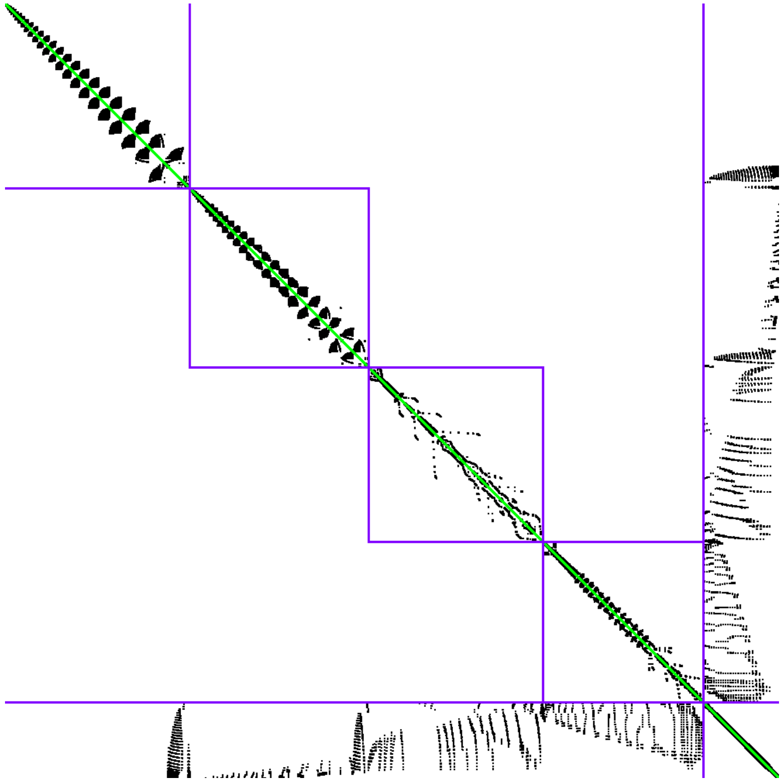


Schur complement

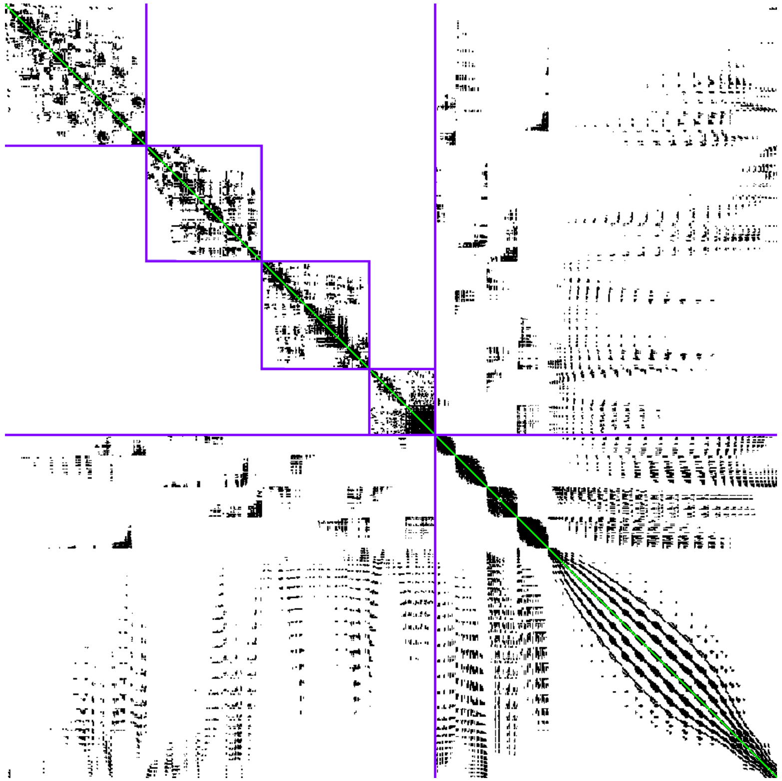


Doubly-Bordered Block-Diagonal Form

First level

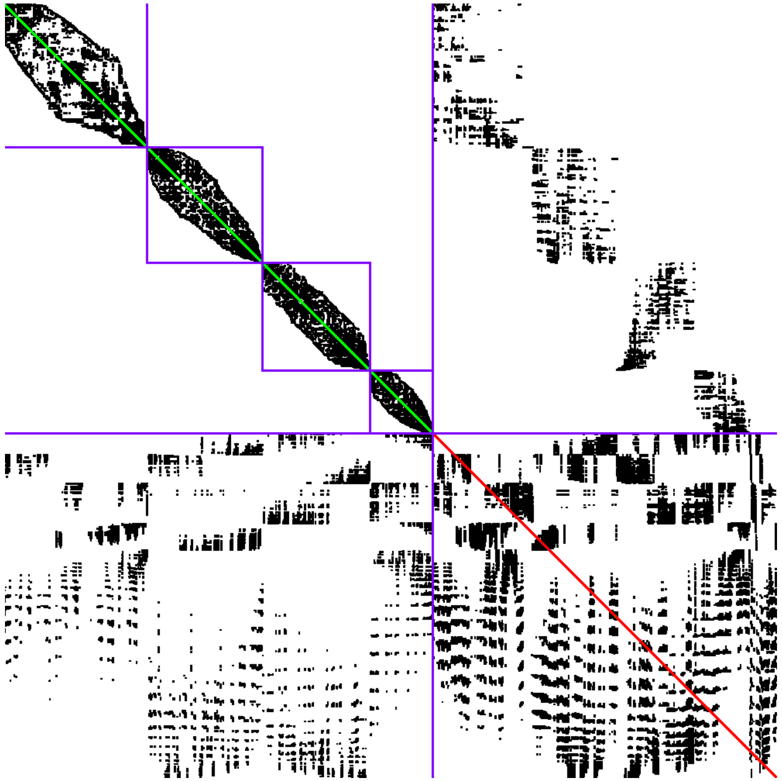


Schur complement

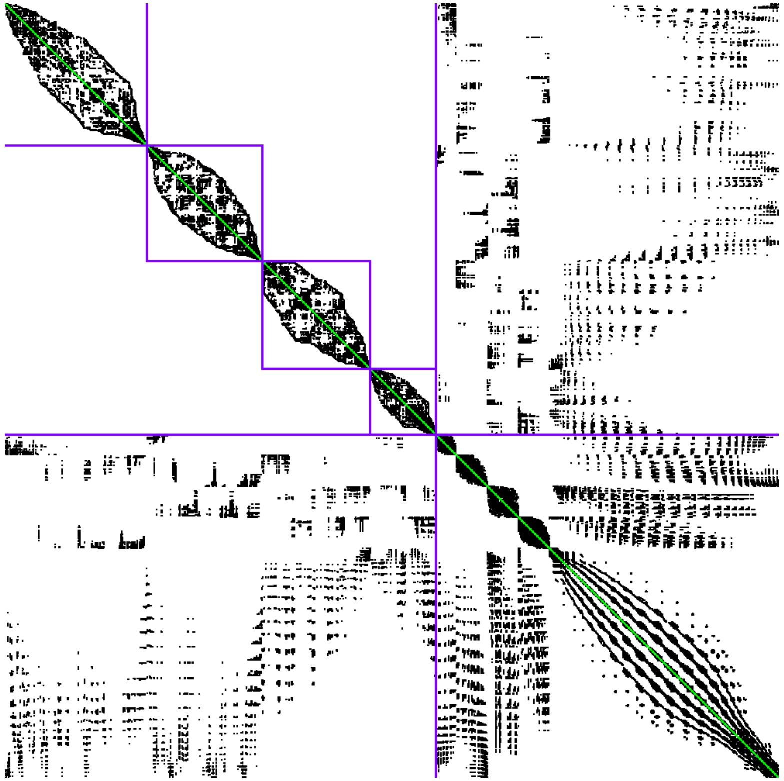


SBBD vs DBBD forms

SBBD Schur (no diagonal)



DBBD Schur (has diagonal)

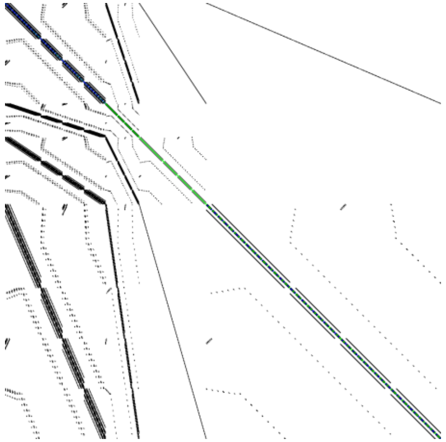


Ordering and Scaling

- Each successive level is **reordered** and **rescaled**:
  - **Reorder** to maximize diagonal product. **(famous MC64)**
    - partially **sequential** ☹
    - may apply **in parallel** to independent blocks, but **reduce** Schur quality.
  - **Rescale** into I-dominant matrix.
  - **Reorder** symmetrically with bandwidth or fill-in reduction algorithm in each block.

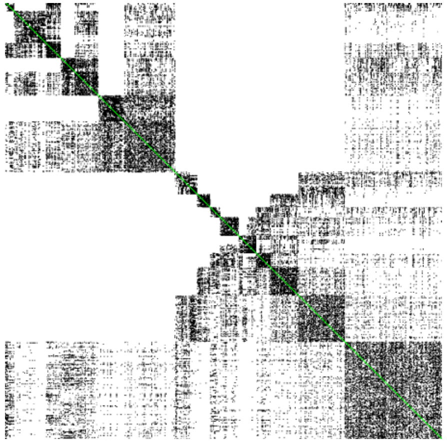
Preprocessing methods

Initial System

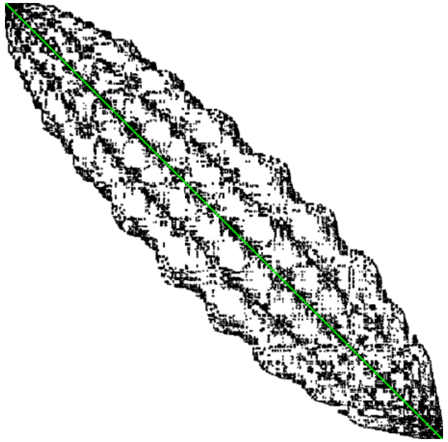


Bandwidth reduction

- Complete **blood coagulation** problem.
- 13 equations per cell.



Fill-in reduction (METIS)

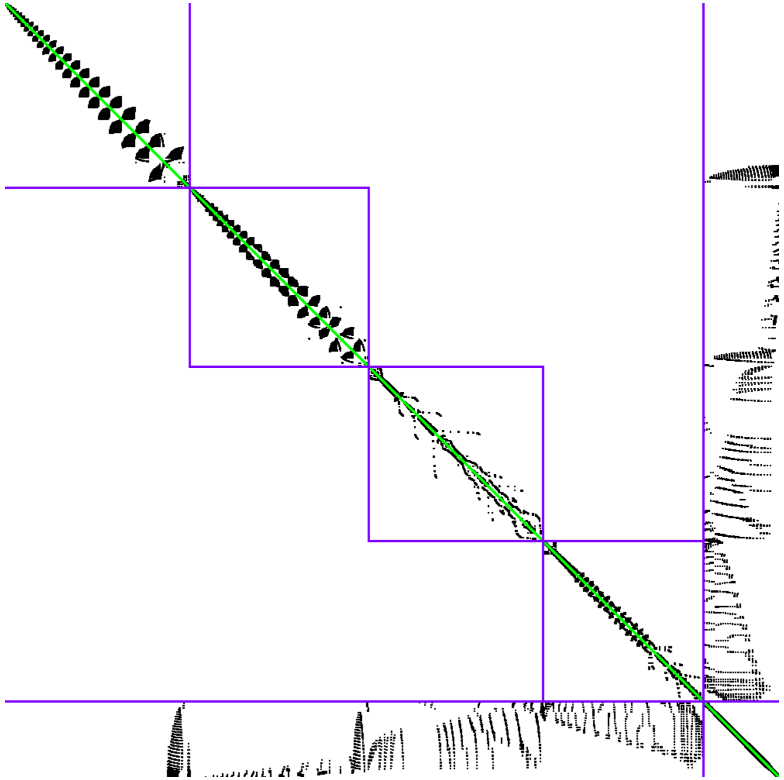


Bandwidth reduction

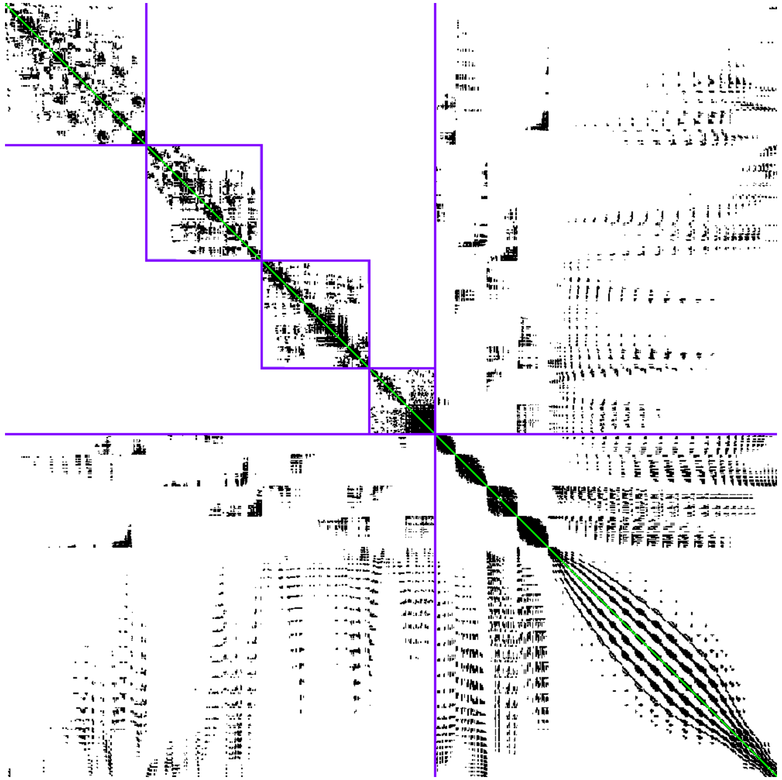
- **Navier-Stokes** problem only.
- 4 equations per cell.
- Fill-in reduction (METIS) is 4 times slower than bandwidth reduction (RCM) algorithm.
- Quite dense matrix graph, big separator.

Doubly-Bordered Block-Diagonal Form (initial)

First level

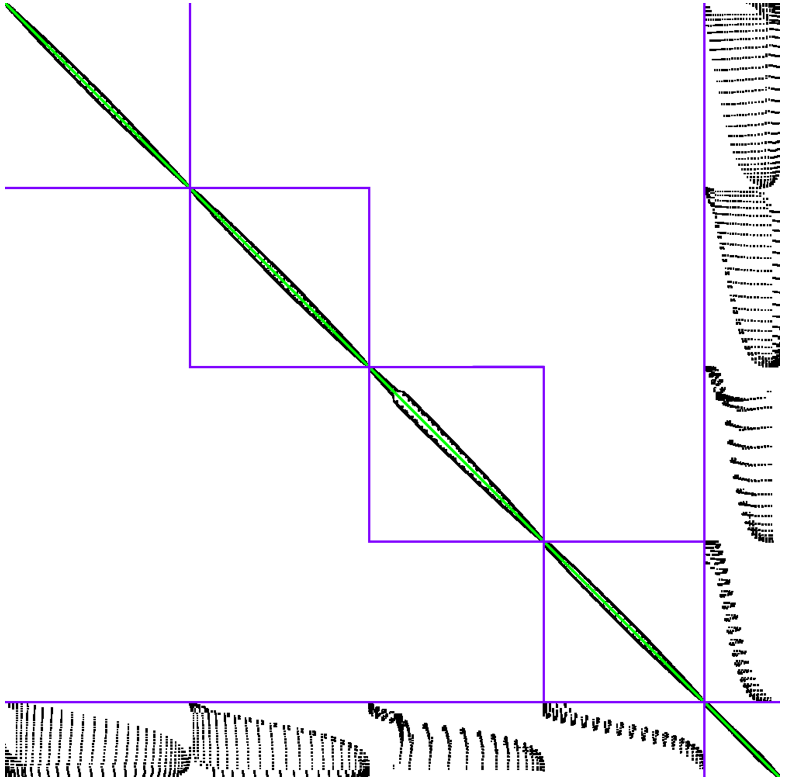


Schur complement

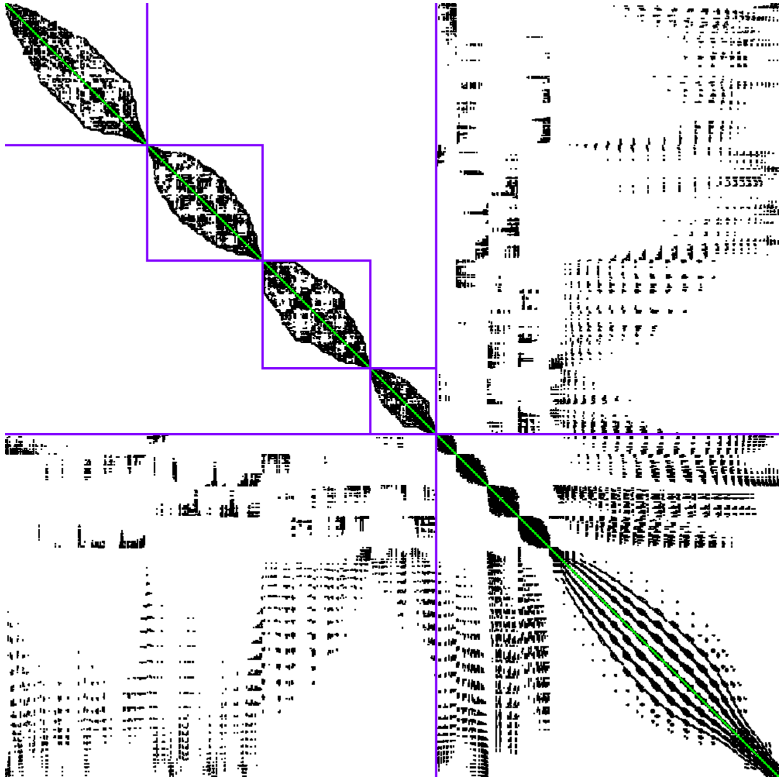


Doubly-Bordered Block-Diagonal Form (RCM)

First level

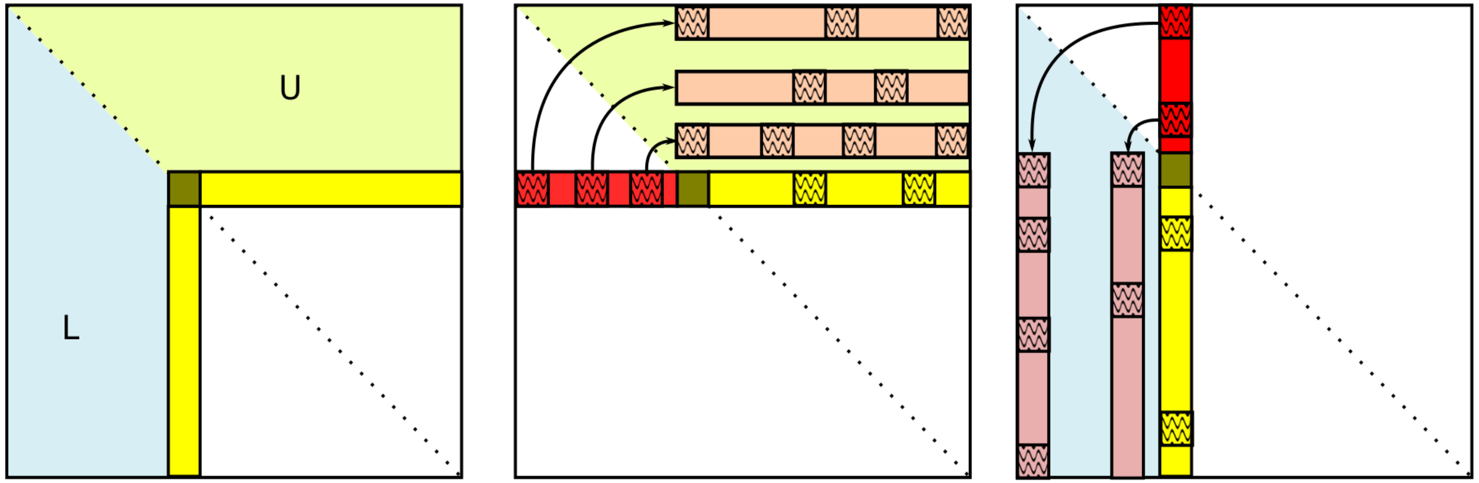


Schur complement



Second-order Crout Incomplete LU

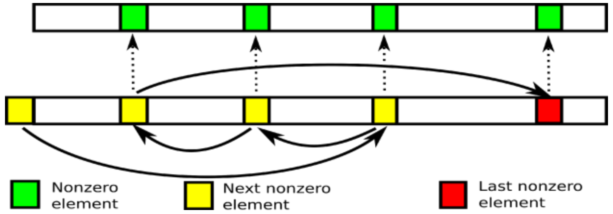
- Dual-threshold dropping:
  - **τ**^{2} for factorization.
  - **τ** for iterations.
- Running condition estimation:
  - **κ** = max(|L^{-1}|,|U^{-1}|)
  - **τ**/**κ**=const tuning.
  - Limit growth of **κ**.



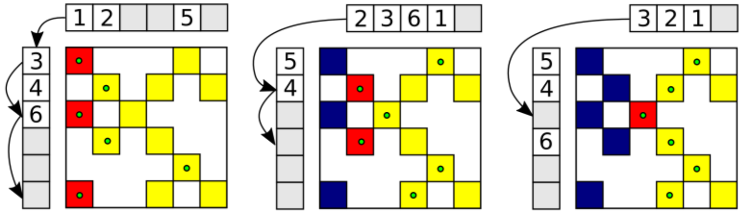
U-factor elimination

L-factor elimination

Dense row accumulator:



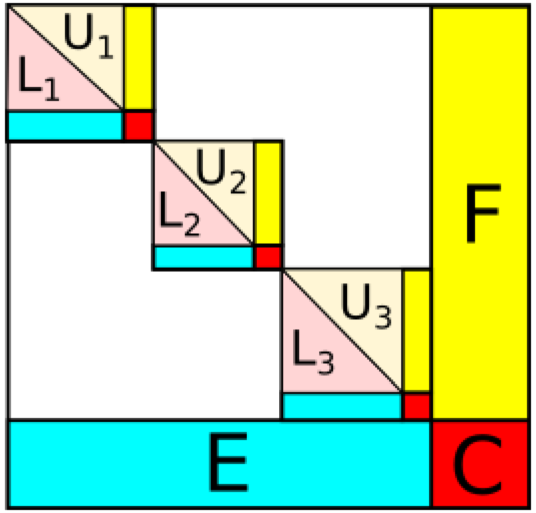
Transposed matrix traversal:



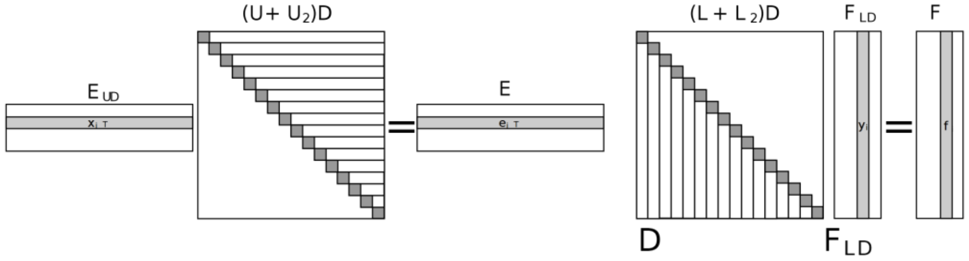
Schur Complement

- Part that leads to growth of **κ** is accumulated in **C:**
  - system reordering after factorization.
- Next level system is the Schur complement:
  - $\mathbf{S} = \mathbf{C} - \mathbf{E}\,(DU)^{-1}\,D(LD)^{-1}\mathbf{F}$.
  - Requires forward and backward substitution with sparse right hand side.
  - Fill-in control is critical.
- **Parallel ☺**

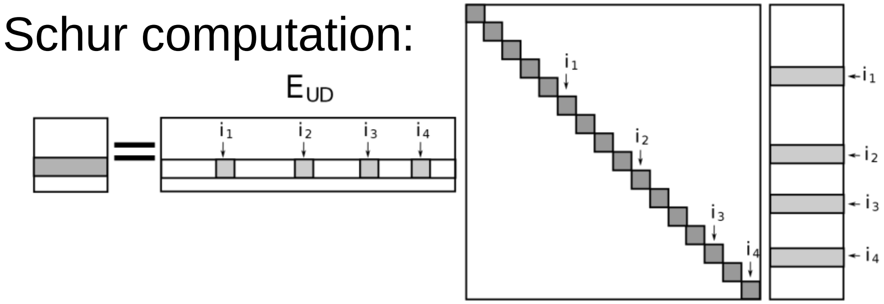
Partially factorized DBBD matrix:



Computation of operators:



Schur computation:



Analogy to the Algebraic Multigrid

- **Coarse** system should contain the **largest** **error** of the **smoothe**r.
- Condition estimation reveals the **error** in the **smoother** and provides the ***coarse-fine splitting*** of the system.
- Ideal prolongation $P=(-EB^{-1},I)$ and restriction $R=(-FB^{-1},I)^T$.
  - (not satisfied by the present method).
- Schur complement corresponds to the **coarse** system.

- **Universal but much more computationally complex.**
  - (definitely not linear computational complexity)

Results

100x100x100 Poisson problem

threads	1	1	2	4	8	16
T_{tot}	**42.7**	47	29.2 (1.6x)	20.2 (2.3x)	13.4 (3.5x)	10.8 (4.3x)
iters	**17**	23	27	25	27	29
T_{iter}	**6**	7.8	4.77 (1.6x)	3.2 (2.4x)	2 (3.9x)	1.64 (4.8x)
levels	**1**	2	5	6	7	7
pivots	**0**	40564	63588	85689	109995	142988
T_{prec}	**36.2**	38.7	23.9 (1.6x)	16.5 (2.3x)	10.8 (3.6x)	8.6 (4.5x)
T_{ord}	**0.5 (1.4%)**	0.7 (2%)	2.6 (11%)	2.4 (15%)	2.3 (21%)	2.55 (29%)
T_{mpt}	**0.3 (0.8%)**	0.4 (1%)	0.4 (1.6%)	0.4 (2.4%)	0.41 (3.7%)	0.44 (5%)
T_{snd}	- (-%)	- (-%)	1.9 (8%)	1.9 (11%)	1.8 (16.6%)	2.04 (24%)
T_{rcm}	**0.2 (0.6%)**	0.4 (1%)	0.3 (1.3%)	0.17 (1%)	0.1 (0.9%)	0.06 (0.8%)
T_{sc}	**0.9 (2.5%)**	1 (2.7%)	0.5 (2.1%)	0.33 (2%)	0.2 (1.9%)	0.17 (1.9%)
T_{fact}	**34.2 (95%)**	12.7 (23%)	6.3 (26.4%)	4 (24%)	2.48 (23%)	1.32 (15%)
T_{schur}	- (-%)	23.7 (71%)	13.9 (58.2%)	9.2 (55%)	5.35 (49%)	4.04 (47%)

Black-oil problem with 1M cells

threads	1	1	2	4	8	16
T_{tot}	**46.3**	56.6	44.7 (1.3x)	37.2 (1.5x)	30.5 (1.9x)	29.4 (1.9x)
iters	**19**	30	30	32	29	35
T_{iter}	**11.3**	25	14.2 (1.8x)	10.7 (2.3x)	6.3 (4x)	6.1 (4.1x)
levels	**1**	5	6	8	9	10
pivots	0	167799	201296	278060	355152	458293
T_{prec}	**33.7**	30.2	29.1 (1x)	25.2 (1.2x)	22.9 (1.3x)	22 (1.4x)
T_{ord}	**5.4 (16%)**	5.5 (18%)	13.1 (45%)	13.3 (53%)	12.3 (54%)	11.5 (52%)
T_{mpt}	**2.9 (9%)**	2.8 (9%)	2.4 (8%)	2.4 (9%)	2.3 (10%)	2.4 (11%)
T_{snd}	- (-%)	- (-%)	7.9 (27%)	8 (32%)	7.8 (34%)	7.8 (36%)
T_{rcm}	**2.5 (7%)**	2.7 (9%)	2.8 (10%)	2.9 (11%)	2.3 (10%)	1.2 (5.5%)
T_{sc}	**4.4 (13%)**	4.6 (15%)	2.6 (9%)	1.7 (7%)	1 (4%)	0.8 (4%)
T_{fact}	**20.3 (60%)**	9.4 (31%)	5.2 (18%)	3 (12%)	3.1 (13%)	2.4 (11%)
T_{schur}	- (-%)	7.1 (23%)	5.2 (18%)	4.8 (19%)	4.3 (19%)	5.2 (24%)

Single-level algorithm

Conclusions and future directions

- Solved memory but not efficiency issues:
  - Sequential algorithm for DBBD form and Schur growth due to increasing deferred pivoting.
- Future directions:
  - Faster algorithm for **DBBD** form.
  - Parallel maximal product transversal.
  - **Block** elimination, **block** pivoting (*reduce deferred pivoting and Schur growth*).
  - **Schur** computation using ideal restriction and prolongation (*more accurate Schur*).
- We are working on a very flexible linear solvers framework S^3M to address multiphysics problems:
  - Konshin, I., Terekhov, K.: ***Sparse System Solution Methods for Complex Problems*. In International Conference on Parallel Computing Technologies,** (2021, September): 53-73. Springer, Cham.

Supported by RSF grant №21-71-20024

Thank you for your attention

Contacts

- KIRILL.TEREHOV@GMAIL.COM

Links

- WWW.INMOST.ORG
- WWW.INMOST.RU