Mathematical Methods in Fluid Dynamics and Simulation of Giant Oil and Gas Reservoirs

3-5 September 2012
Swissotel The Bosphorus, Istanbul, Turkey

Fast and robust solvers for pressure systems on the GPU

Kees Vuik

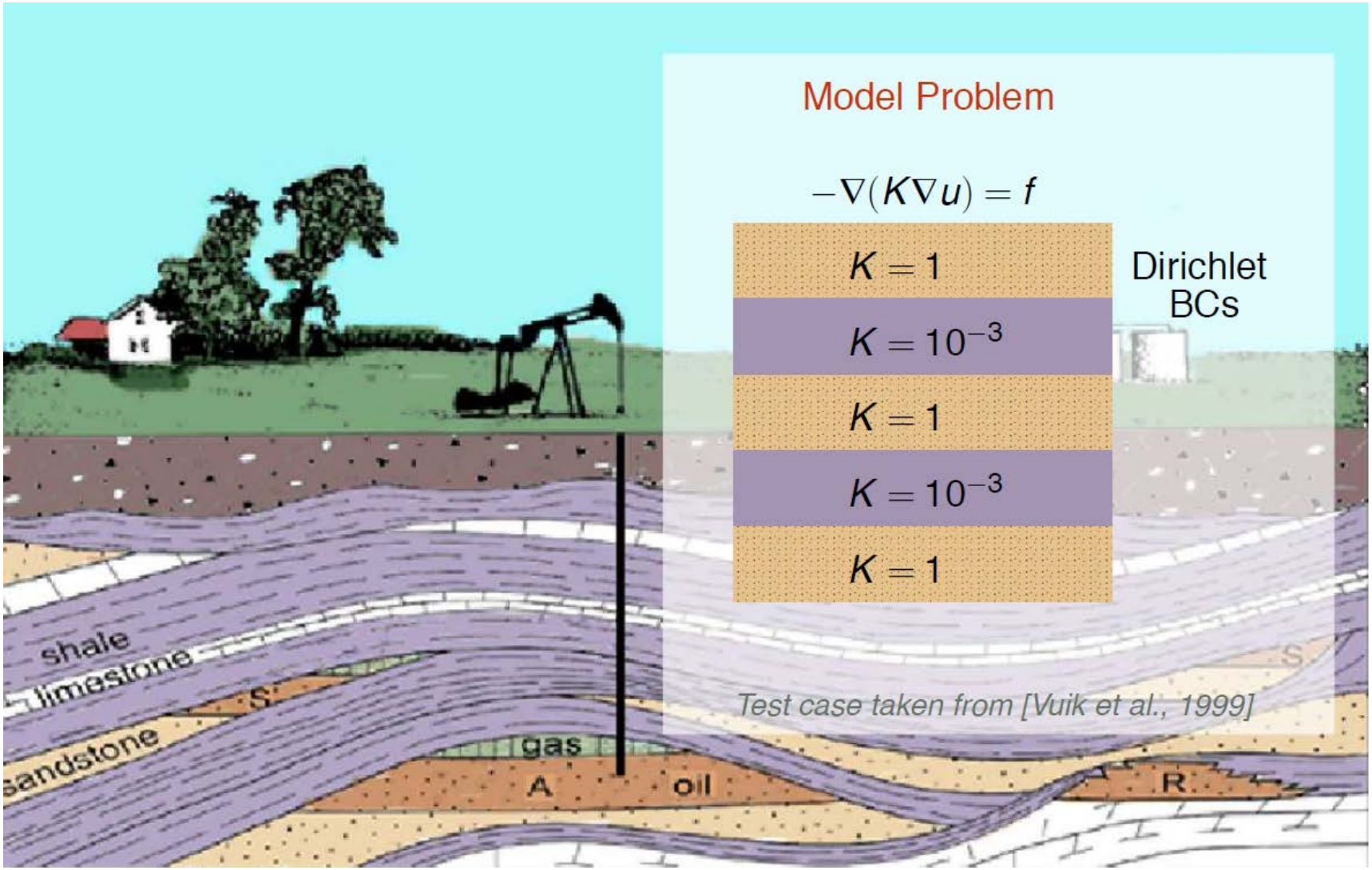
Delft University of Technology

http://ta.twi.tudelft.nl/users/vuik/

Contents

1. Large Jumps
2. Large Grids
3. Large Computers
4. Summary

1. Large Jumps



Properties and Applications

$Ax = b$

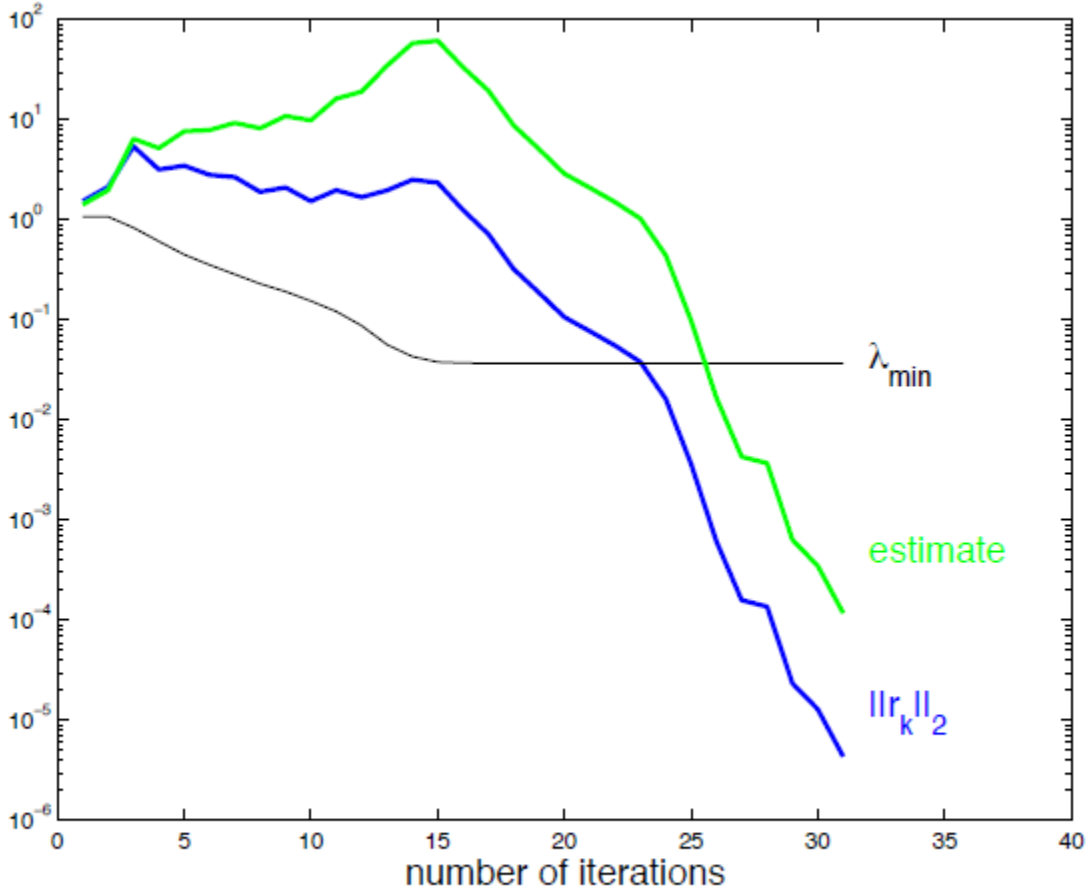
A is sparse and SPD

Condition number of A is large, due to large contrast in permeability

Applications

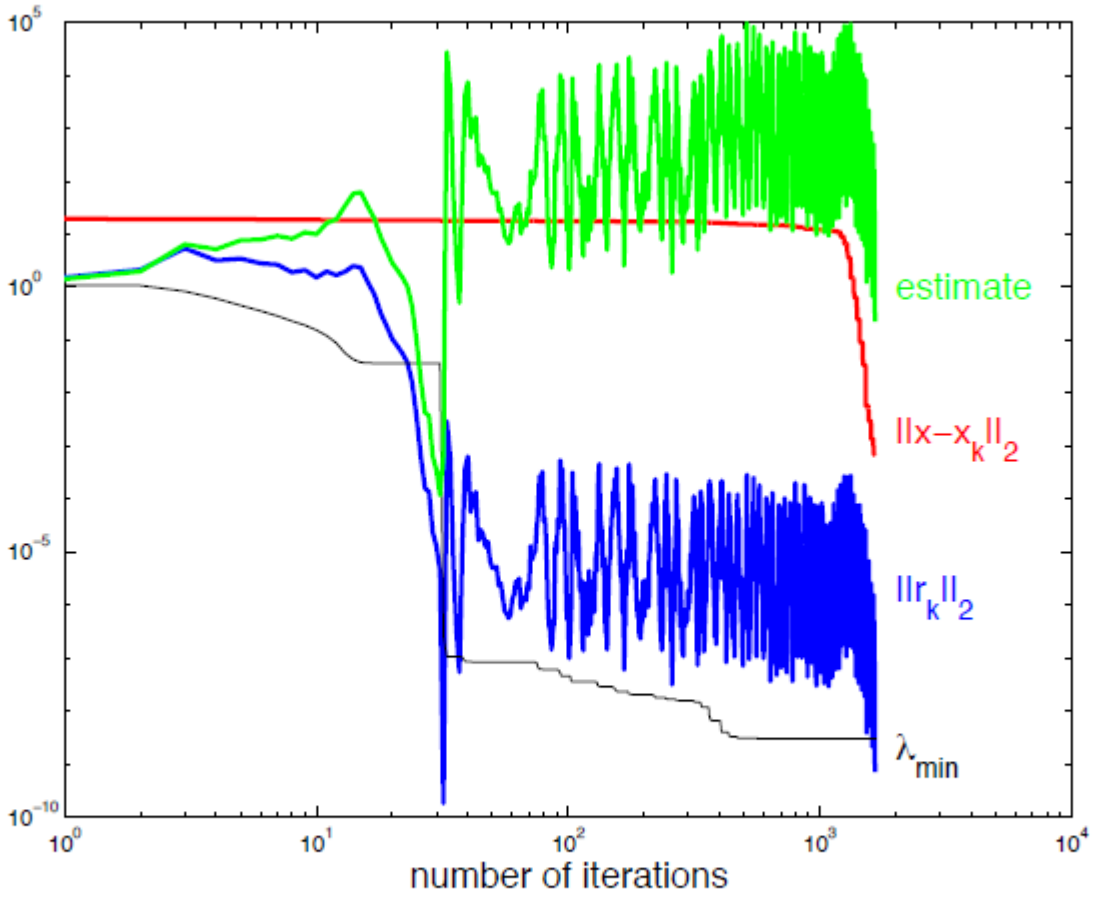
- Reservoir simulations
- Porous media flow
- Fictitious domain methods

Convergence of CG



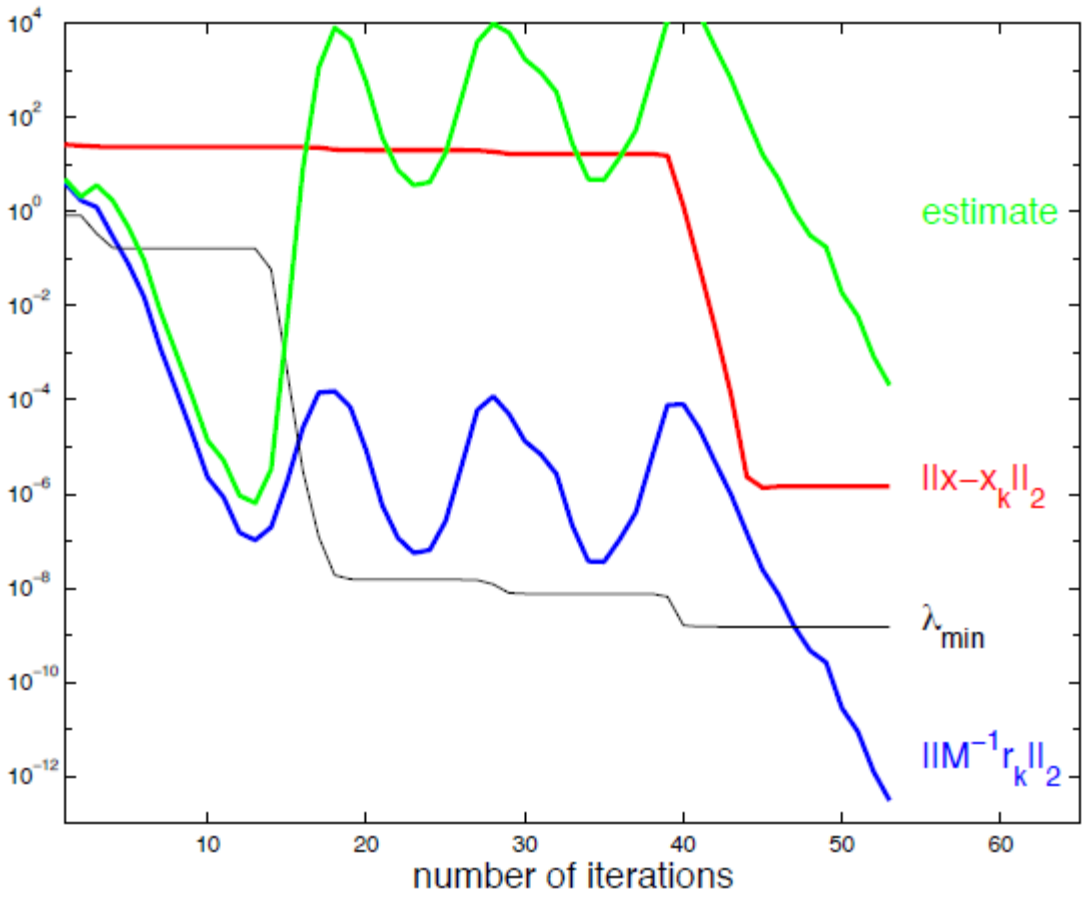
Convergence behavior of CG without preconditioning

Convergence of CG



Convergence behavior of CG without preconditioning

Convergence of CG



Convergence behavior of ICCG

Deflated ICCG

Idea: remove the bad eigenvectors from the error/residual.

Krylov	Ar
Preconditioned Krylov	$M^{-1}Ar$
Block Preconditioned Krylov	$\sum_{i=1}^{m}(M_i^{-1})Ar$
Block Preconditioned Deflated Krylov	$\sum_{i=1}^{m}(M_i^{-1})PAr$

DICCG

DICCG

$$k = 0,\ \hat{r}_0 = Pr_0,\ p_1 = z_1 = L^{-T}L^{-1}\hat{r}_0;$$

while $\|\hat{r}_k\|_2 > \varepsilon$ **do**

$$k = k + 1;$$

$$\alpha_k = \frac{(\hat{r}_{k-1}, z_{k-1})}{(p_k, PAp_k)};$$

$$x_k = x_{k-1} + \alpha_k p_k;$$

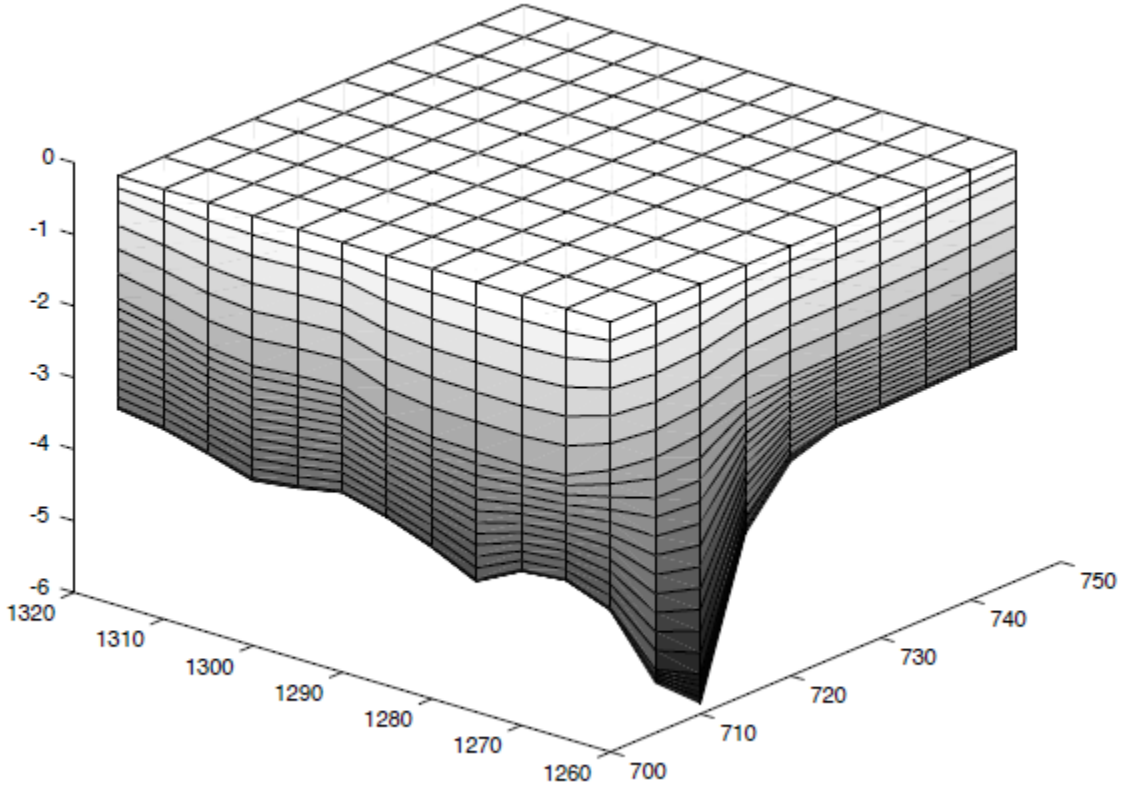
$$\hat{r}_k = \hat{r}_{k-1} - \alpha_k PAp_k;$$

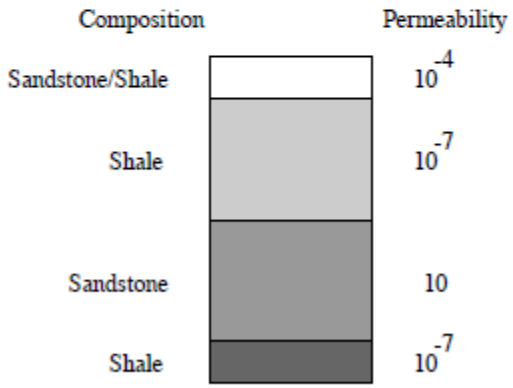
$$z_k = L^{-T}L^{-1}\hat{r}_k;$$

$$\beta_k = \frac{(\hat{r}_k, z_k)}{(\hat{r}_{k-1}, z_{k-1})}; \qquad p_{k+1} = z_k + \beta_k p_k;$$

end while

Geometry oil flow problem





Results oil flow problem

Varying σ_{shale}

σ	ICCG		DICCG	
	λ_{min}	iter	λ_{min}	iter
10^{-3}	$1.5 \cdot 10^{-2}$	26	$6.9 \cdot 10^{-2}$	20
10^{-5}	$2.2 \cdot 10^{-4}$	59	$7.7 \cdot 10^{-2}$	20
10^{-7}	$2.3 \cdot 10^{-6}$	82	$7.7 \cdot 10^{-2}$	20

Varying accuracy

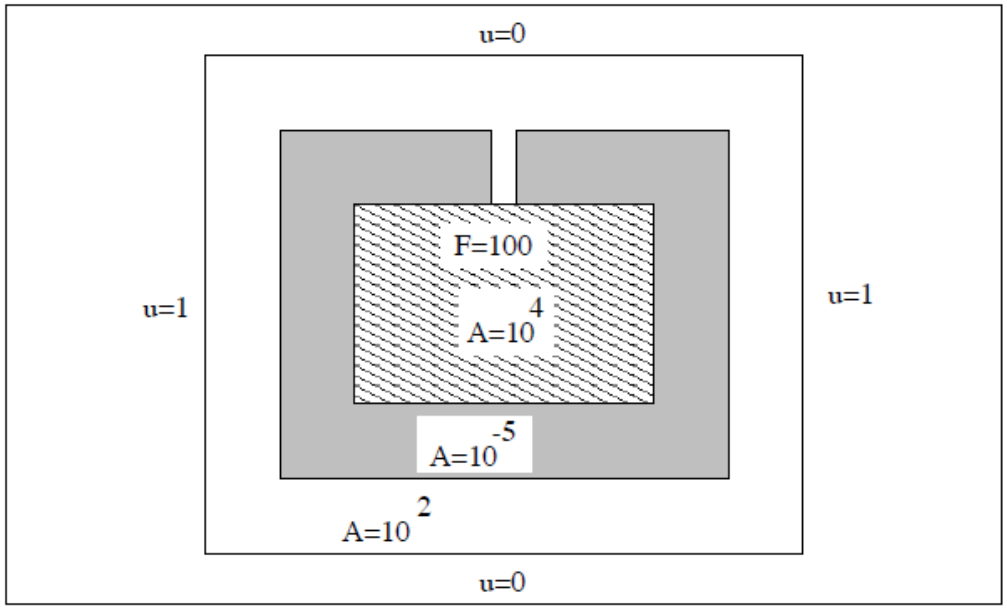
accuracy	ICCG		DICCG	
	iter	CPU	iter	CPU
10^{-5}	82	18.9	20	6.3
10^{-3}	78	18.0	12	4.1
10^{-1}	75	17.2	2	1.2

A groundwater flow problem

The pressure in groundwater satisfies the equation:

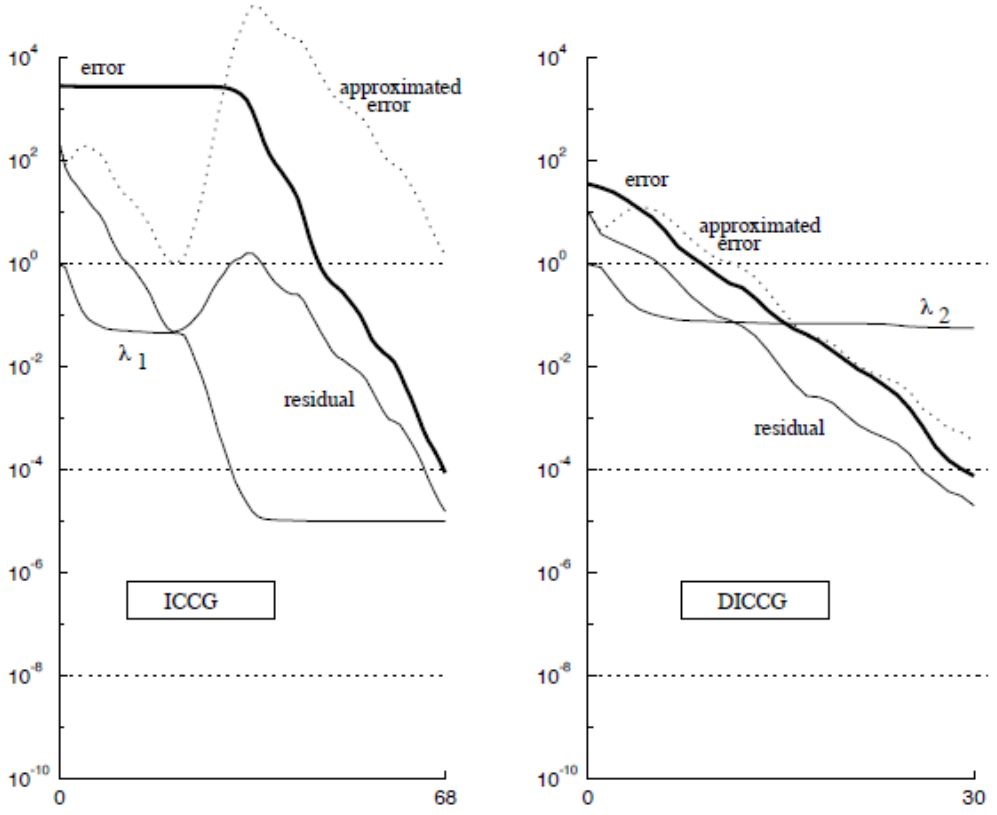
$$-\nabla \cdot (A \nabla u) = F,$$

where the coefficients and geometry of the problem are:



A groundwater flow problem

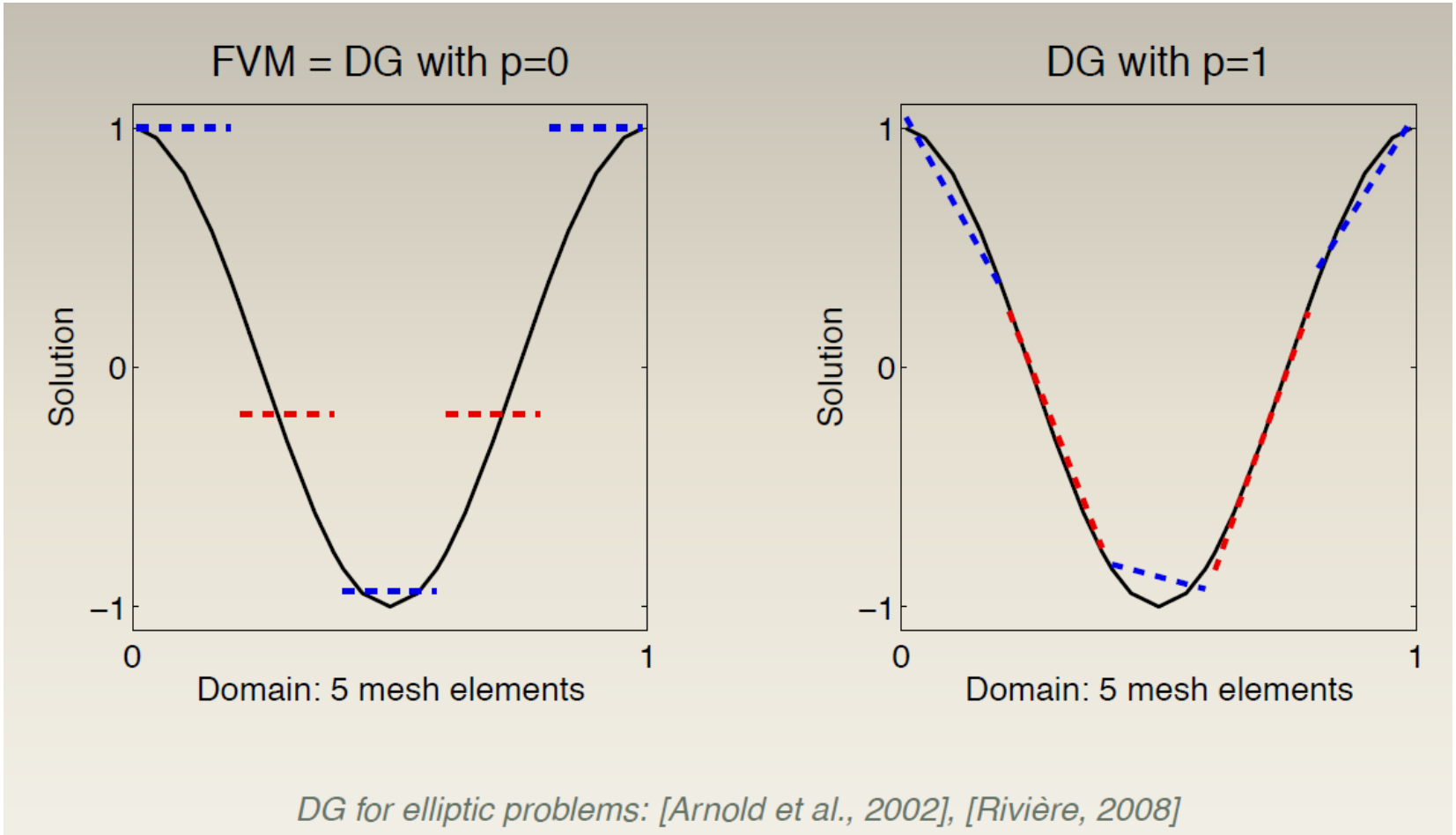
The low permeable layer ($A = 10^{-5}$) and the jump in permeabilities between the two sand sections lead to a 'small' eigenvalue.



2. Large grids (PhD, Paulien van Slingerland)

DG Methods

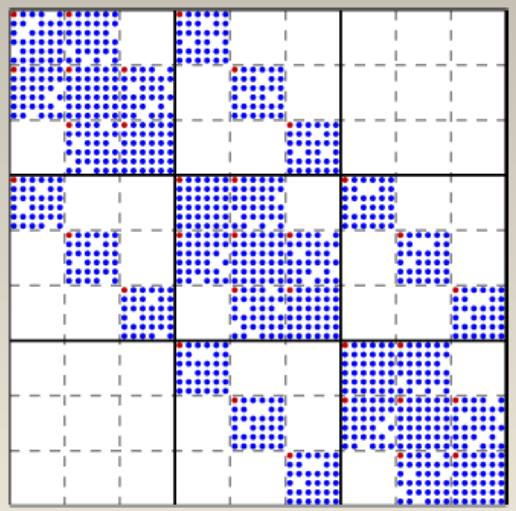
DG methods are like FVM, but then based on piecewise polynomials

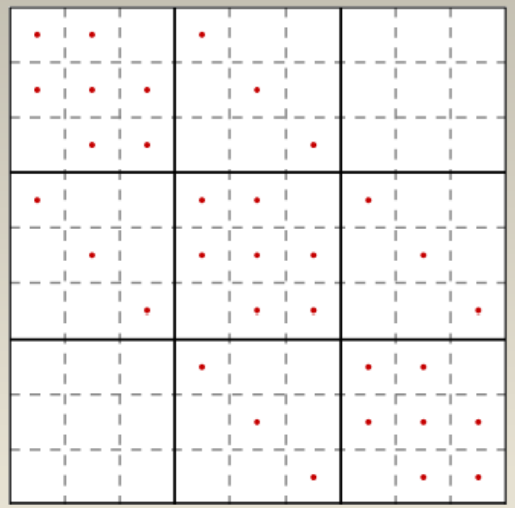


DG for elliptic problems: [Arnold et al., 2002], [Rivière, 2008]

Coarse corrections

The main idea is to speed up CG using coarse corrections based on $p = 0$





DG matrix A with $p > 0$

DG matrix RAR^T with $p = 0$

$$A^{-1} \approx Q := \underbrace{R^T}_{\text{prolongation}} (RAR^T)^{-1} \underbrace{R}_{\text{restriction}}$$

Original idea of spectral multigrid: [Rønquist and Patera, 1987]

Deflation variant

We can switch to deflation by simply skipping a smoothing step

The result z of applying two-level deflation to a vector r:

$$z^{(1)} := \omega M^{-1} r, \qquad \text{apply smoother } \omega M^{-1}$$

$$z^{(2)} := z^{(1)} + Q(r - Az^{(1)}), \qquad \text{apply coarse correction } Q$$

~~$z := z^{(2)} + \omega M^{-T}(r - Az^{(2)})$, apply smoother ωM^{-T}~~

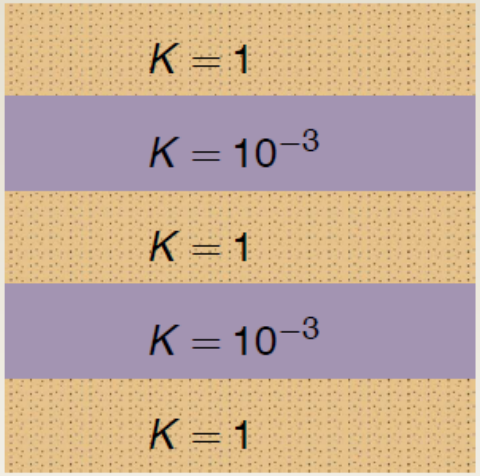
Requirement: ~~$M + M^T - \omega A$ is SPD~~ M is SPD
assuming we pre-process the CG start vector once:

$$x_0 \mapsto Qb + (I - AQ)^T x_0,$$

The result is equivalent to a CG method with an SPD preconditioner: [Tang et al., 2009]

Layered problem

degree	p=2				p=3			
mesh	$N=20^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=160^2$
size A	2400	9600	38400	153600	4000	16000	64000	256000
Jacobi	975	1264	1567	2314	1295	1490	1921	3110
block Jacobi (BJ)	243	424	788	1285	244	425	697	1485
two-level prec., 2x BJ	46	43	43	44	55	56	56	57
two-level defl., 1x BJ	43	45	45	46	47	48	48	48

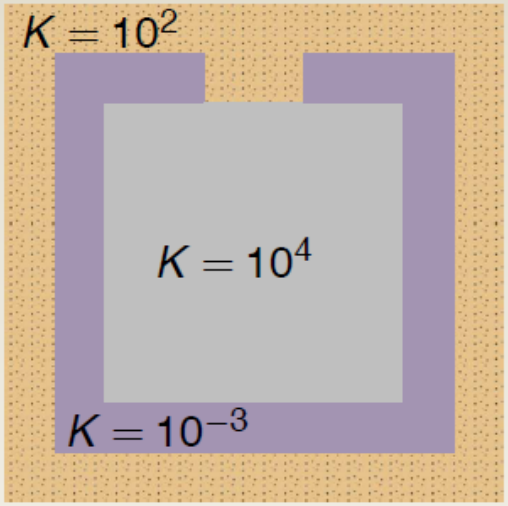


CG stopping criterion: $\frac{||b-Ax_k||_2}{||b||_2} \leq 10^{-6}$

Diagonal-scaling is applied beforehand

Groundwater flow

degree	p=2				p=3			
mesh	$N=20^2$	$N=40^2$	$N=80^2$	$N=160^2$	$N=20^2$	$N=40^2$	$N=80^2$	$N=160^2$
two-level prec. ($\omega = 1$)	53	54	52	52	63	67	68	68
two-level prec. ($\omega = 0.7$)	36	38	38	38	39	41	42	42
two-level defl. ($\omega = 1$)	52	54	54	54	58	59	59	60



Test case taken from [Vuik et al., 2001]

Variational Boussinesq model (VBM) as proposed by Gert Klopman

Linearized VBM equations:

$$\frac{\partial \zeta}{\partial t} + \nabla \cdot (\zeta \mathbf{U} + h\nabla\varphi - h\mathcal{D}\nabla\psi) = 0,$$

$$\frac{\partial \varphi}{\partial t} + \mathbf{U} \cdot \nabla\varphi + g\zeta = -P_s,$$

$$\mathcal{M}\psi + \nabla \cdot (h\mathcal{D}\nabla\varphi - \mathcal{N}\nabla\psi) = 0.$$

3. Large computers (GPU, MSc, Martijn de Jong)

- Equidistant rectangle grid (not mandatory)
- Finite volume method (FVM) for space
- Leapfrog method for time integration

After discretization:

$$S\vec{\psi} = \mathbf{b},$$

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = L\mathbf{q} + \mathbf{f}.$$

Solve them consecutively in real-time

3. Large computers (GPU)

Matrix S is given by a 5-point stencil:

$$\begin{bmatrix} 0 & -\frac{\Delta x}{\Delta y}\overline{\mathcal{N}_N} & 0 \\ -\frac{\Delta y}{\Delta x}\overline{\mathcal{N}_W} & \frac{\Delta x}{\Delta y}\overline{\mathcal{N}_N} + \frac{\Delta y}{\Delta x}\overline{\mathcal{N}_E} + \Delta x \Delta y \mathcal{M}_C + \frac{\Delta x}{\Delta y}\overline{\mathcal{N}_S} + \frac{\Delta y}{\Delta x}\overline{\mathcal{N}_W} & -\frac{\Delta y}{\Delta x}\overline{\mathcal{N}_E} \\ 0 & -\frac{\Delta x}{\Delta y}\overline{\mathcal{N}_S} & 0 \end{bmatrix}.$$

Matrix S is:

- real-valued, sparse (5-point, pentadiagonal)
- diagonally dominant (not very strong for small mesh sizes)
- symmetric positive definite (SPD)
- quite large (in the order of millions by millions)

The RRB-solver

The RRB-solver:

- is a PCG-type solver (preconditioned conjugated gradients)
- uses as preconditioner: the RRB-method

RRB stands for “Repeated Red-Black”.

The RRB-method determines an incomplete factorization:

$$S = LDL^T + R \quad \Longrightarrow \quad M = LDL^T \approx S$$

The RRB-solver

As the name RRB reveals: multiple levels

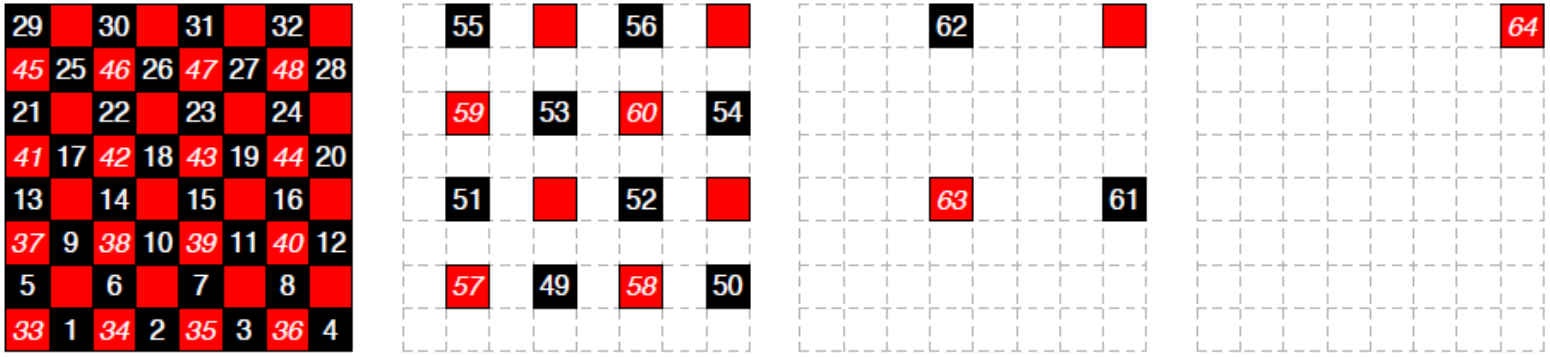
Therefore the RRB-solver has good scaling behaviour (Multigrid)

Method of choice because:

- shown to be robust for all of MARIN's test problems
- solved all test problems up to 1.5 million nodes within 7 iterations(!)

Special ordering

An 8×8 example of the RRB-numbering process



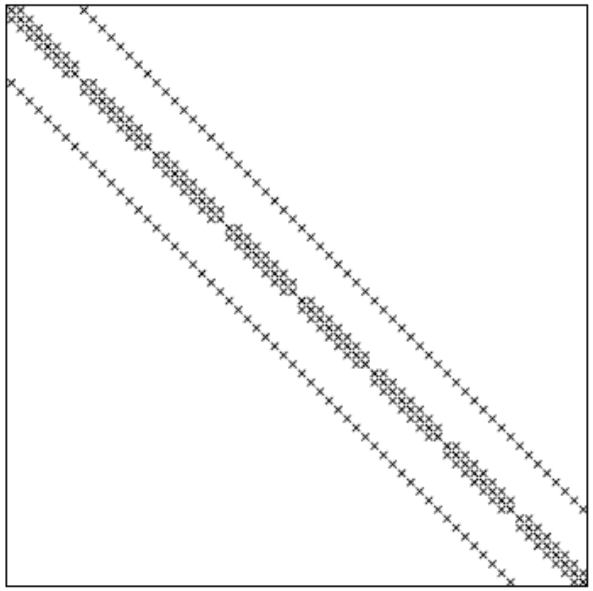
(1) (2) (3) (4)

All levels combined:

29	55	30	62	31	56	32	64
45	25	46	26	47	27	48	28
21	59	22	53	23	60	24	54
41	17	42	18	43	19	44	20
13	51	14	63	15	52	16	61
37	9	38	10	39	11	40	12
5	57	6	49	7	58	8	50
33	1	34	2	35	3	36	4

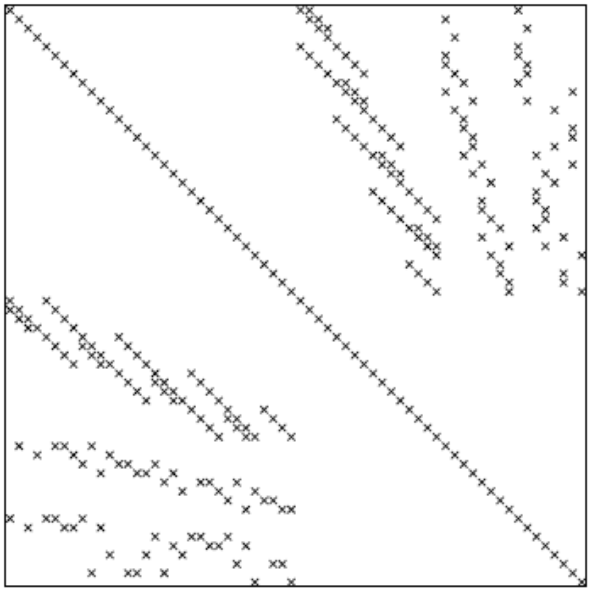
Special ordering

Effect on sparsity pattern of matrix S:



Lexicographic

becomes

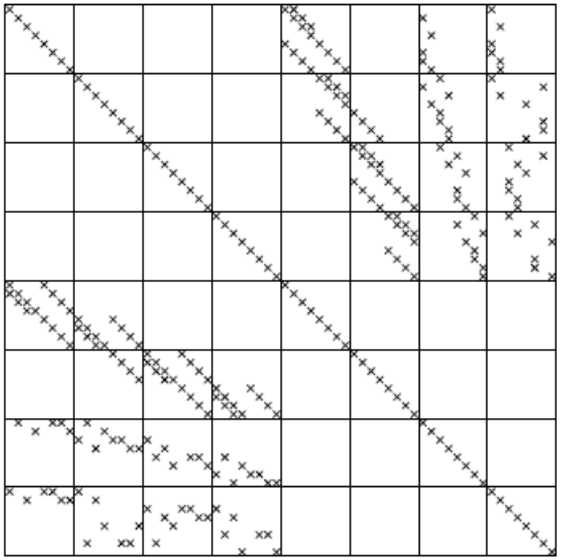


RRB-numbering

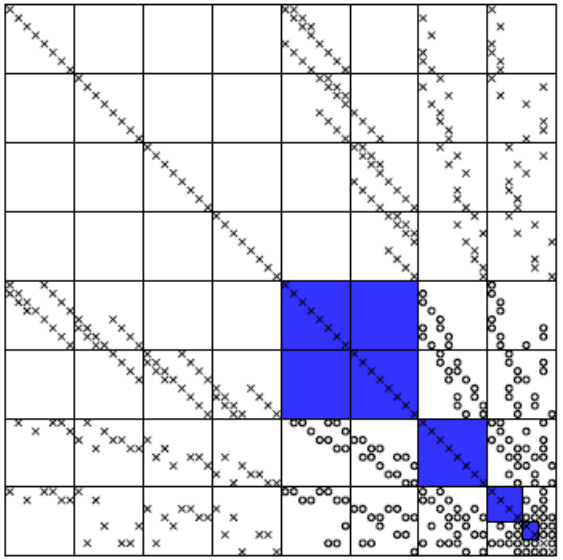
Special ordering

Sparsity pattern of matrix S versus $L + D + L^T$

(recall preconditioner $M = LDL^T$)



becomes



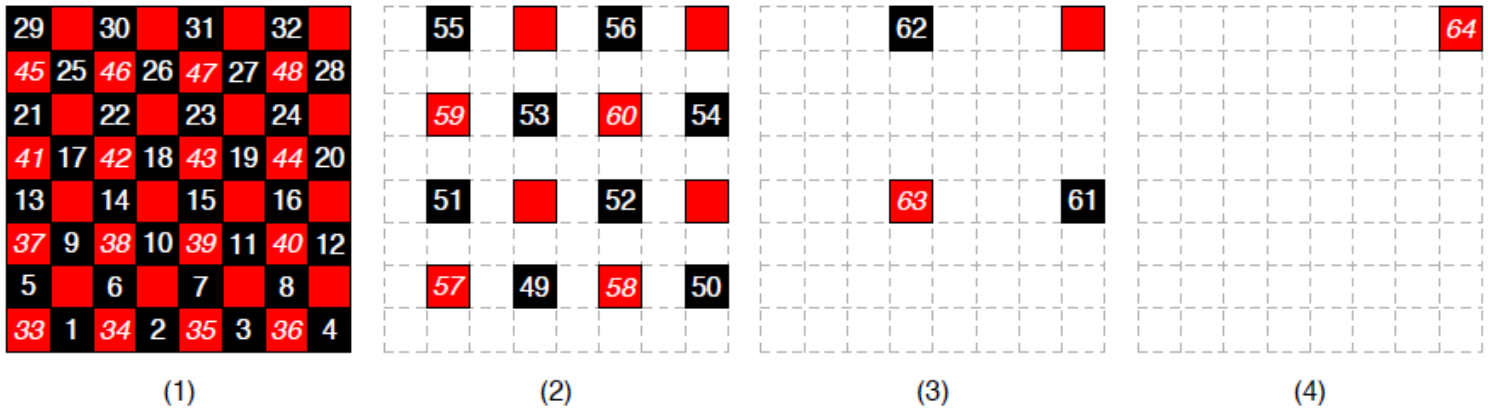
In the blue shaded areas fill-in has been dropped (lumping)

CUDA implementation

Besides the typical Multigrid issues such as idle cores on the coarsest levels, in CUDA the main problem was getting “coalesced memory transfers”.

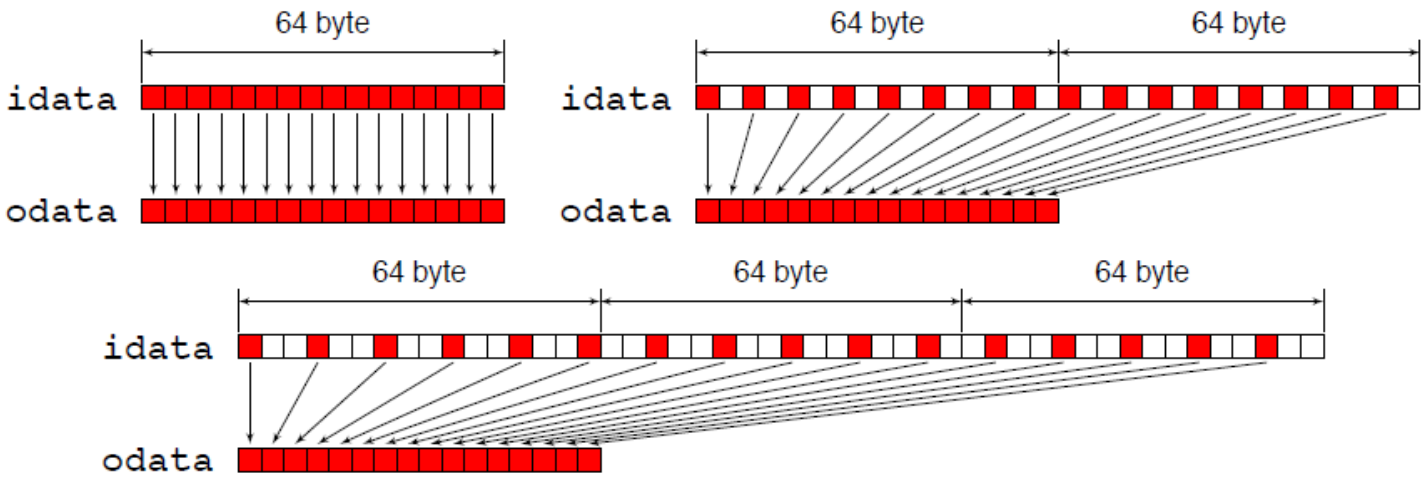
Why is that?

Recall the RRB-numbering: the number of nodes becomes 4× smaller on every next level:



CUDA implementation

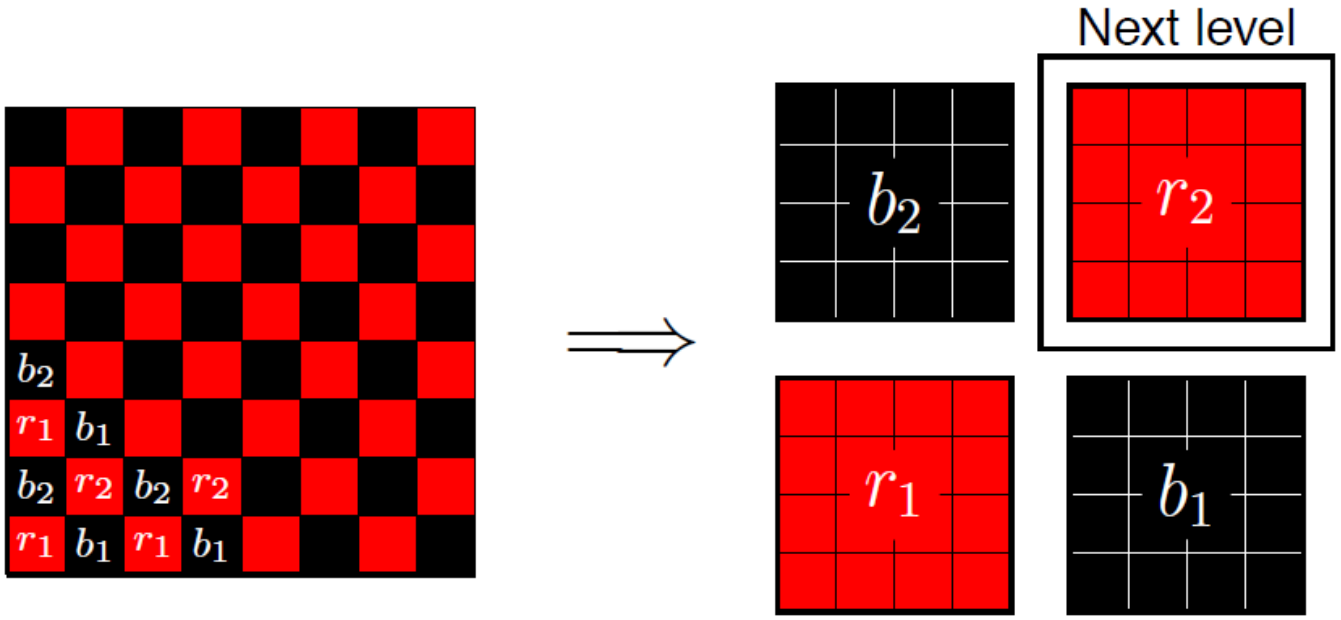
- Data is read from and written to the device's global memory via 32-, 64- of 128-byte memory transfers
- Example: reading data with a stride



CUDA implementation

New storage scheme: $r_1/r_2/b_1/b_2$

Nodes are divided in four groups:



CUDA implementation

The *r1/r2/b1/b2*-storage scheme

• is applied on every next coarser level till the point that the remaining level is smaller than 32 × 32 elements; the last levels are solved in one go on 1 streaming multiprocessor (SM) exploiting the benefits of cache

• almost comes for free (only at the beginning and ending of CG we have some overhead due to reordering of the data)

• allows for coalesced memory read and write operations throughout the entire CG algorithm which yields optimal throughput

Results

Kernel throughput up to 250 GB/s (thanks to cache)

• Solver speed up is up to 30× for realistic problems of 1.5 million nodes and up to 40× for even larger problems (> 2048 × 2048 nodes)

• Time needed? Merely 10 milliseconds for 7 CG-iterations (vs. 300 ms for C++)

• The fast CUDA solver allows real-time simulation

• Also the RRB-preconditioner can be constructed in real-time and hence varying bathymetry across time is supported

Summary

- Building blocks for fast and robust solvers for pressure systems on the GPU are given
- Deflation can reduce the condition number, number of iterations, and CPU time considerably
- High accuracy DG methods can greatly reduce the grid size
- Deflation type solvers lead to scalable solvers for DG problems
- RRB solver leads to scalable convergence for Poisson type problems
- Clever reordering leads to speed up of a factor 40 on the GPU

References

- C. Vuik, A. Segal and J.A. Meijerink, J. Comp. Phys., 152, pp. 385-403, 1999.
- J. Frank and C. Vuik, SIAM Journal on Scientific Computing, 23, pp. 442–462, 2001
- R. Nabben and C. Vuik, SIAM Journal on Scientific Computing, 27, pp. 1742-1759, 2006
- P. van Slingerland and C. Vuik Spectral two-level deflation for DG: a preconditioner for CG that does not need symmetry Delft University of Technology Delft Institute of Applied Mathematics, Report 11-12 http://ta.twi.tudelft.nl/nw/users/vuik/papers/Sli11V.pdf
- Martijn de Jong Developing a CUDA solver for large sparse matrices for MARIN Master Thesis, Delft University of Technology, 2012 http://ta.twi.tudelft.nl/nw/users/vuik/numanal/jong afst.pdf