Iteratieve methoden

Kees Vuik
c.vuik@math.tudelft.nl

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

Delft University of Technology

TU Delft

Inhoud

1. Iteratieve methoden
2. Preconditioneringen
3. Deflatie
4. Parallelle methoden

TU Delft

1. Iteratieve methoden

LSQR	Paige and Saunders
GMRES	Saad and Schultz
CGS	Sonneveld
Bi-CGSTAB	Van der Vorst and Sonneveld

TU Delft

1. Iteratieve methoden

LSQR	Paige and Saunders
GMRES	Saad and Schultz
CGS	Sonneveld
Bi-CGSTAB	Van der Vorst and Sonneveld
GCR	Eisenstat, Elman and Schultz
GMRESR	Van der Vorst and Vuik

TU Delft

GCR and GMRESR

Preconditioned GCR

$k = 0,\ r_0 = b - Ax_0$

while $\|r_k\|_2 > \varepsilon$ **do**

  $k = k + 1$, solve $Ms_k = r_k$

  $v_k = As_k$

  for $j = 1, \ldots k - 1$ **do**

    $\alpha = (v_k, v_j), v_k = v_k - \alpha v_j, s_k = s_k - \alpha s_j$

  end

  $v_k = v_k / \|v_k\|_2, s_k = s_k / \|v_k\|_2$

  $x_k = x_{k-1} + (r_{k-1}, v_k) s_k$

  $r_k = r_{k-1} - (r_{k-1}, v_k) v_k$

end while

GCR and GMRESR

Preconditioned GMRESR

$k = 0,\ r_0 = b - Ax_0$

while $\|r_k\|_2 > \varepsilon$ **do**

$k = k + 1$, solve with GMRES(m): $AMs_k = r_k$

$v_k = As_k$

for $j = 1, \ldots k - 1$ **do**

$\alpha = (v_k, v_j), v_k = v_k - \alpha v_j, s_k = s_k - \alpha s_j$

end

$v_k = v_k / \|v_k\|_2, s_k = s_k / \|v_k\|_2$

$x_k = x_{k-1} + (r_{k-1}, v_k) s_k$

$r_k = r_{k-1} - (r_{k-1}, v_k) v_k$

end while

Preconditioneringen

- Standaard iteratieve methoden
- Multigrid
- ILU
  - ordening
  - fill in
  - lumping
  - blok varianten
- Sparse Approximate Inverse

TU Delft

Toepassingen

- Poisson op een vierkant met EDM

Toepassingen

- Poisson op een vierkant met EDM
- Poisson: EEM, EVM, gebied, randvoorwaarden, coëfficiënten
- Convectie-diffusie

TU Delft

Toepassingen

- Poisson op een vierkant met EDM
- Poisson: EEM, EVM, gebied, randvoorwaarden, coëfficiënten
- Convectie-diffusie
- Ontkoppelde Navier-Stokes
- Gekoppelde Navier-Stokes

Toepassingen

- Poisson op een vierkant met EDM
- Poisson: EEM, EVM, gebied, randvoorwaarden, coëfficiënten
- Convectie-diffusie
- Ontkoppelde Navier-Stokes
- Gekoppelde Navier-Stokes
- Mechanica problemen
- Helmholtz/Maxwell problemen
- . . .

2. Preconditioneringen

If A is SPD the Choleski decomposition LL^T exists.
Only the lower triangular part of A is stored in memory.

Incomplete Choleski (IC) decomposition

Compute L which satisfies:

- $l_{ij} = 0$ for $a_{ij} = 0$ and $i > j$
- $(LL^T)_{ij} = a_{ij}$ for $a_{ij} \neq 0$ and $i \geq j$

This preconditioner exists if A is an M-matrix.

$$a_{ij} \leq 0, \;\; j \neq i \text{ and } A^{-1} \geq 0.$$

Break down IC preconditioner

Suppose A is not an M-matrix.

$$l_{ii}^2 = a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2$$

Break down when right-hand side is negative.

Break down IC preconditioner

Suppose A is not an M-matrix.

$$l_{ii}^2 = a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2$$

Break down when right-hand side is negative.

Typical examples

- large convection and central differences
- linear elements with bad triangles
- quadratic elements
- coupled problems: mechanical problems, Navier-Stokes

Solution strategies

1. Use LDL^T and MINRES
2. Replace right-hand side by a small positive number ($a_{ii}/100$)
3. Use IC of $A + \gamma I$ or $A + \gamma\text{diag}(A)$ with a suitable chosen $\gamma > 0$.
4. Construct $\hat{A}$ such that $\hat{A} \approx A$ and $\hat{A}$ is an M-matrix. Use IC of $\hat{A}$.

Lumped IC preconditioners

Suppose A is SPD but not an M-matrix. Construct $\hat{A}$ as follows:

for $i = 1, n$
  $\hat{a}_{ii} = a_{ii}$
  for $j = 1, i-1$
    if $a_{ij} < 0$ **then**
      $\hat{a}_{ij} = a_{ij}$
    else
      $\hat{a}_{ij} = 0$, $\hat{a}_{ii} = \hat{a}_{ii} + a_{ij}$, $\hat{a}_{jj} = \hat{a}_{jj} + a_{ij}$
    end if
  end for
end for

$\hat{A}$ is an M-matrix. Its IC decomposition exists and is used as a preconditioner for $Ax = b$.

Convection diffusion equation

- square domain, diffusion coefficient $\mathbb{D}$
- angle α, central scheme, 100×100 grid

TU Delft

Convection diffusion equation

- square domain, diffusion coefficient $\mathbb{D}$
- angle α, central scheme, 100×100 grid

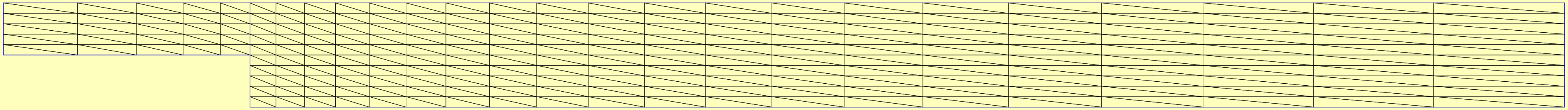
Number of matrix vector products angle is 45^o

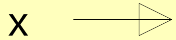
$\mathbb{D}$	Bi	Bi-CGSTAB	GMRES	GMRESR(10)
10^{-3}	42	8	7	7
10^{-4}	nc	52	48	47
10^{-5}	nc	392	207	189
10^{-6}	nc	nc	438	350
10^{-7}	nc	nc	$>$ 500	389

Convergence for lumped ILU is only weakly dependent on the angle

Backward Facing Step

Coupled Navier-Stokes problem





MESH

TU Delft

Backward Facing Step

No convergence without lumping

Backward Facing Step

No convergence without lumping
Stokes

grid	Bi-CGSTAB matvec	CPU	GMRES matvec	CPU	GMRESR matvec	CPU
10×25	484	2.02	1293	6.16	315	1.77
20×50	1398	24.59	497	10.33	485	12.25

TU Delft

Backward Facing Step

No convergence without lumping

Stokes

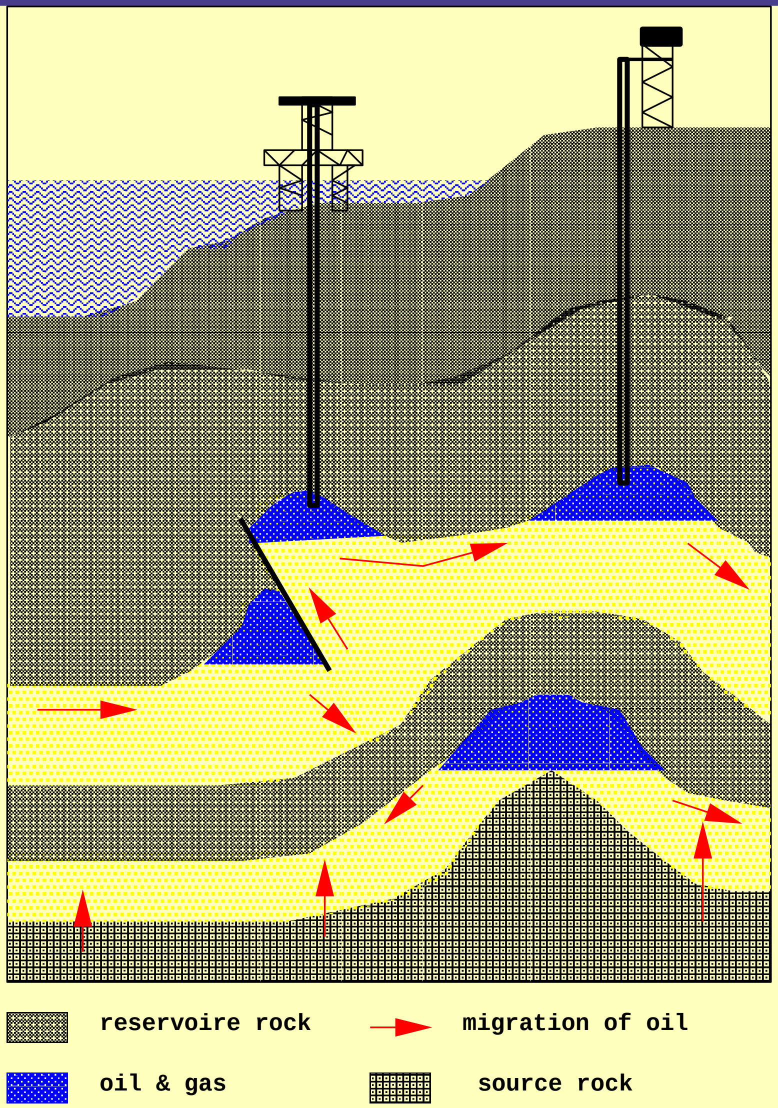
grid	Bi-CGSTAB matvec	CPU	GMRES matvec	CPU	GMRESR matvec	CPU
10×25	484	2.02	1293	6.16	315	1.77
20×50	1398	24.59	497	10.33	485	12.25

Navier-Stokes

grid	Bi-CGSTAB matvec	CPU	GMRES matvec	CPU	GMRESR matvec	CPU
10×25	2138	9.38	7828	37	2195	12.36
20×50	4592	85	11107	228	3895	98

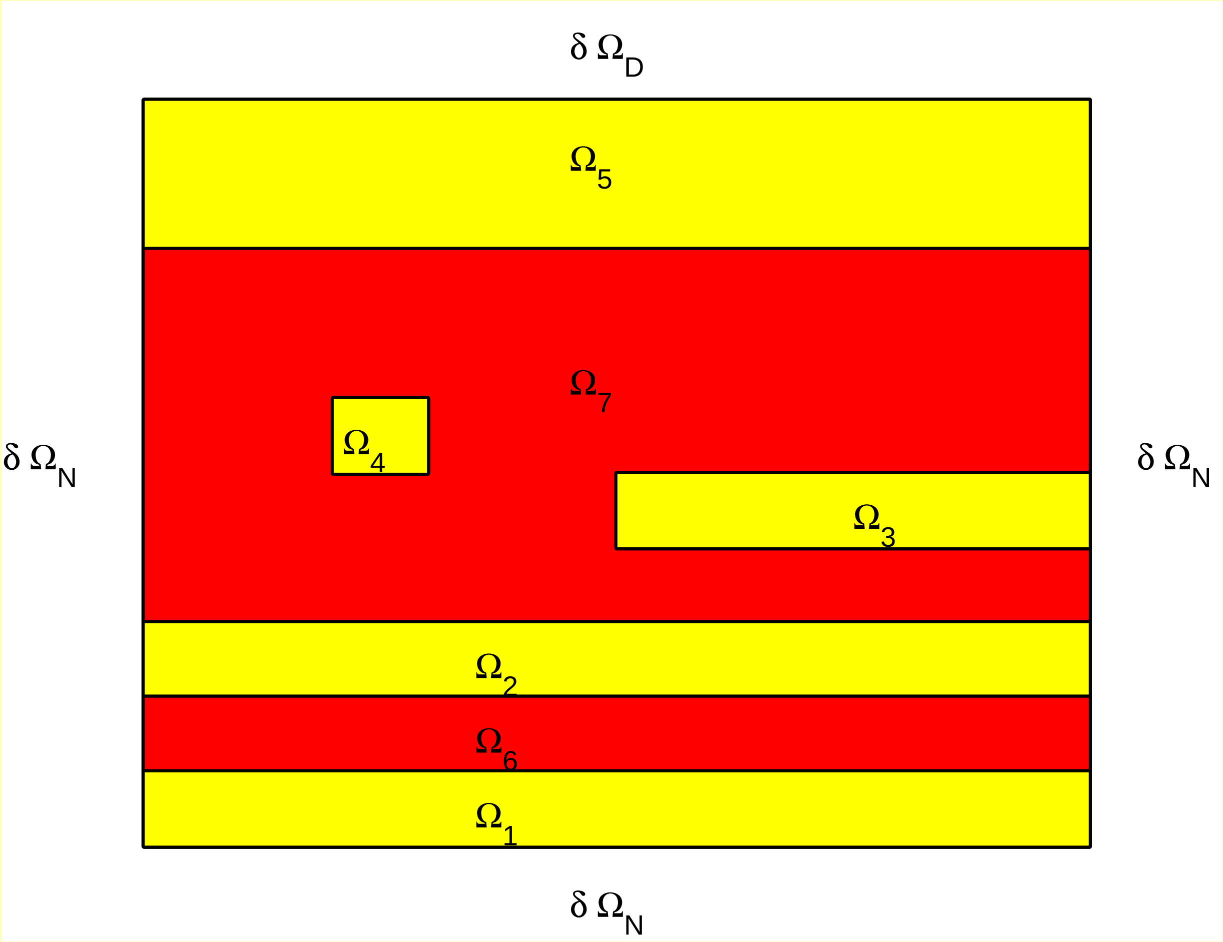
TU Delft

3. Deflatie



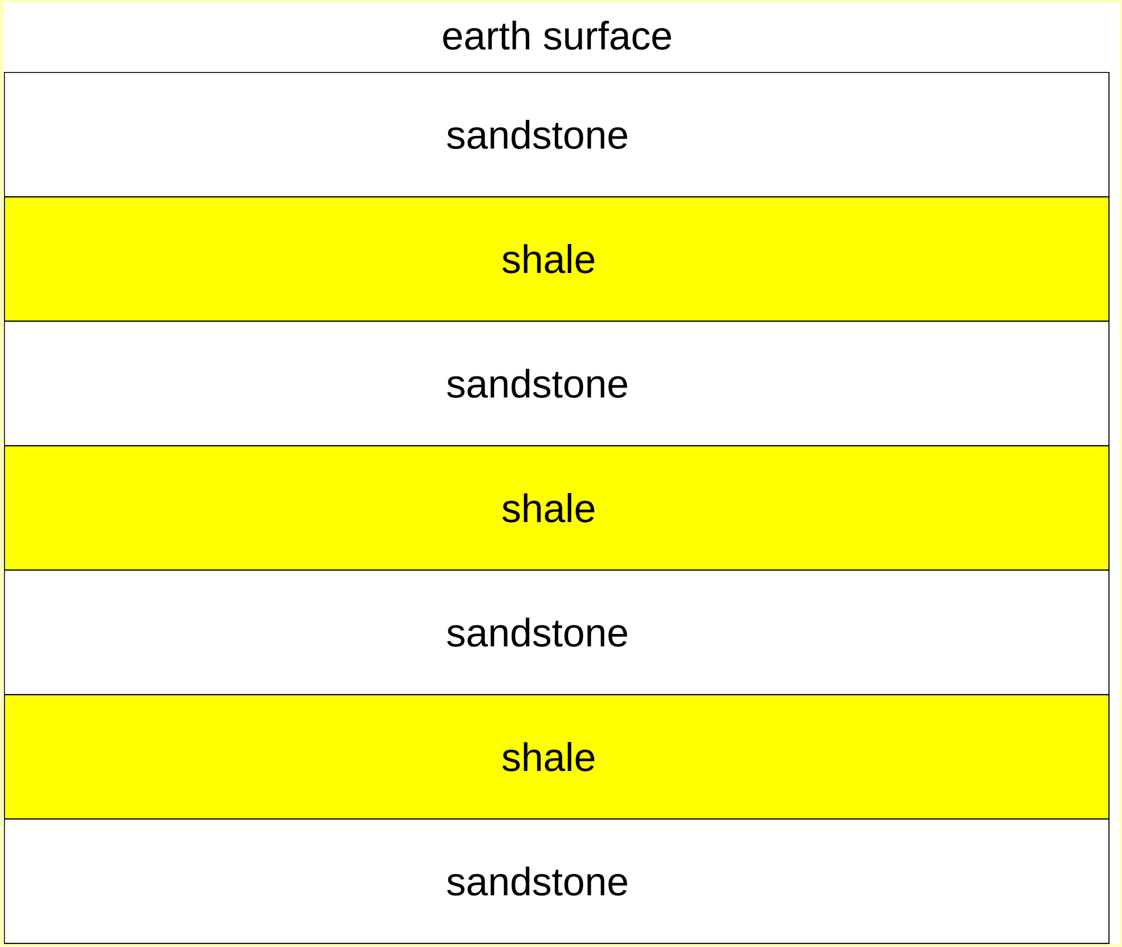
Mathematical model: computation of fluid pressure

$-\text{div}(\sigma \nabla p(x)) = 0$ on Ω, p fluid pressure, σ permeability



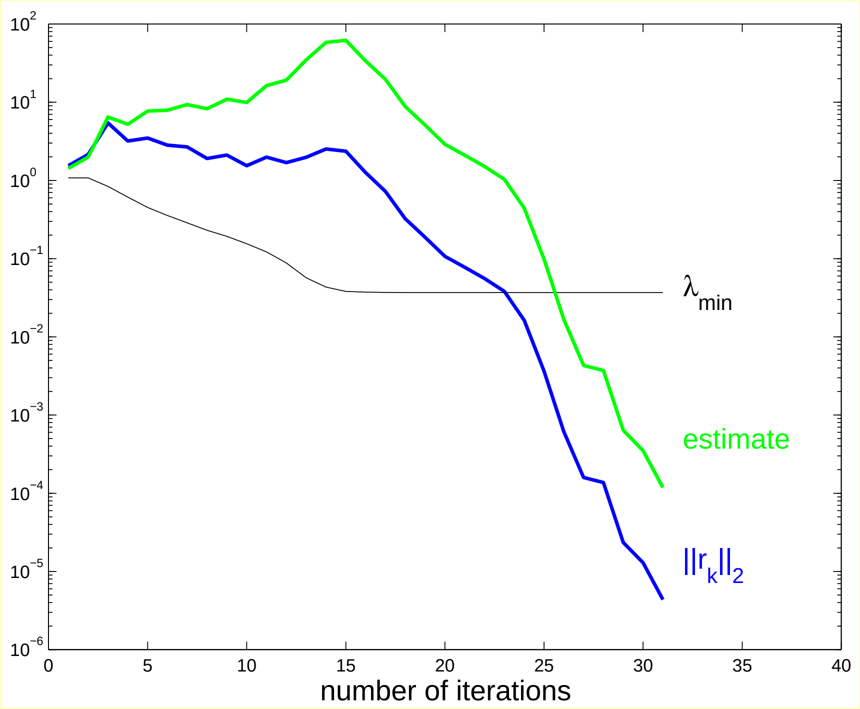
TU Delft

Test problem



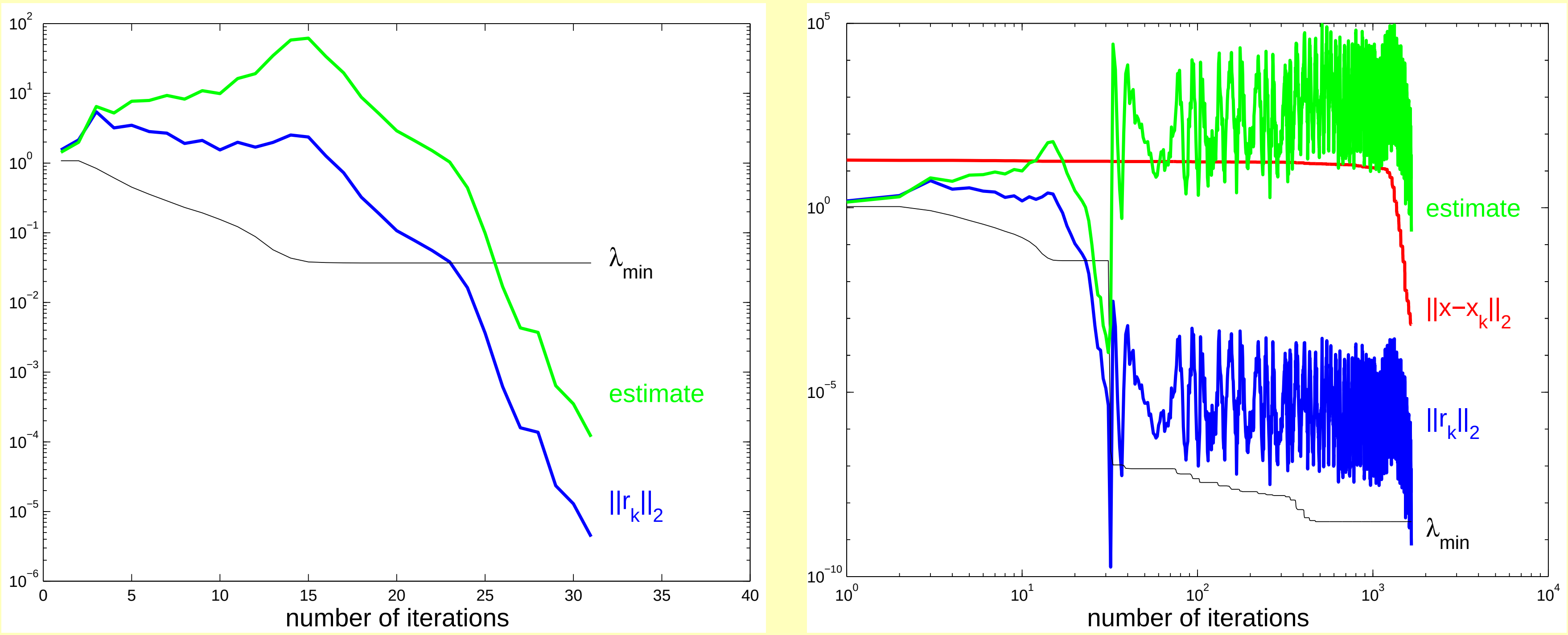
Configuration with 7 straight layers

Convergence CG, without preconditioning



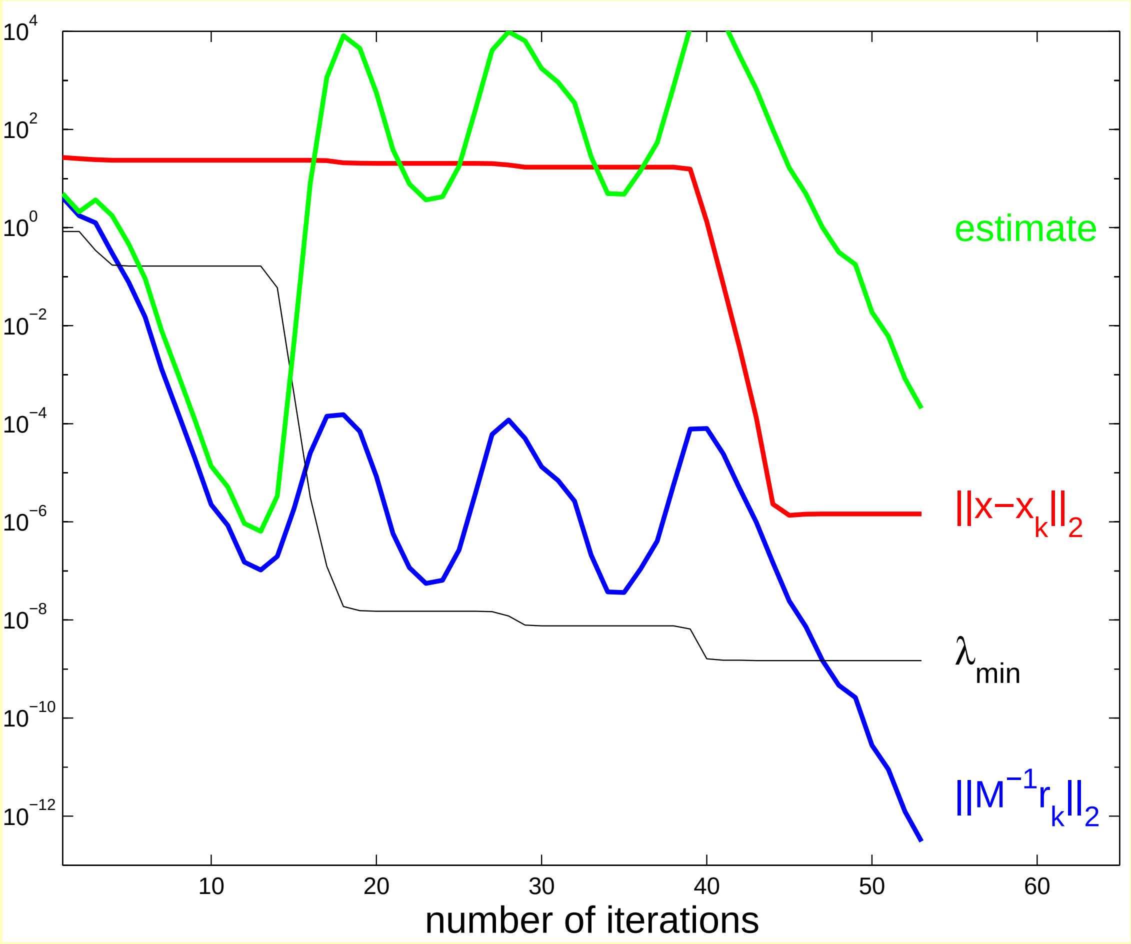
TU Delft

Convergence CG, without preconditioning



TU Delft

Convergence CG, with IC preconditioning



TU Delft

Deflated ICCG

A is SPD, Conjugate Gradients

$$P = I - AZE^{-1}Z^T \text{ with } E = Z^T AZ$$

and $Z = [z_1 ... z_m]$, where $z_1, ..., z_m$ are independent deflation vectors.

Properties

1. $P^T Z = 0$ and $PAZ = 0$
2. $P^2 = P$
3. $AP^T = PA$

Deflated ICCG

$$x = (I - P^T)x + P^T x$$

$(I - P^T)x = ZE^{-1}Z^T Ax = ZE^{-1}Z^T b$ and $AP^T x = PAx = Pb$

TU Delft

Deflated ICCG

$$x = (I - P^T)x + P^T x$$

$$(I - P^T)x = ZE^{-1}Z^T Ax = ZE^{-1}Z^T b \text{ and } AP^T x = PAx = Pb$$

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

TU Delft

Physical deflation vectors

k is number of subdomains

$\Omega_i, i = 1, ..., k^s$ high-permeability subdomains without a Dirichlet b.c.;
$i = k^s + 1, ..., k^h$ remaining high-permeability subdomains

- $z_i = 1$ on $\bar{\Omega}_i$ and $z_i = 0$ on $\bar{\Omega}_j, j \neq i,$
  $i \in \{1, ..., k^s\}$ and $j \in \{1, ..., k^h\}$
- z_i satisfies equation:

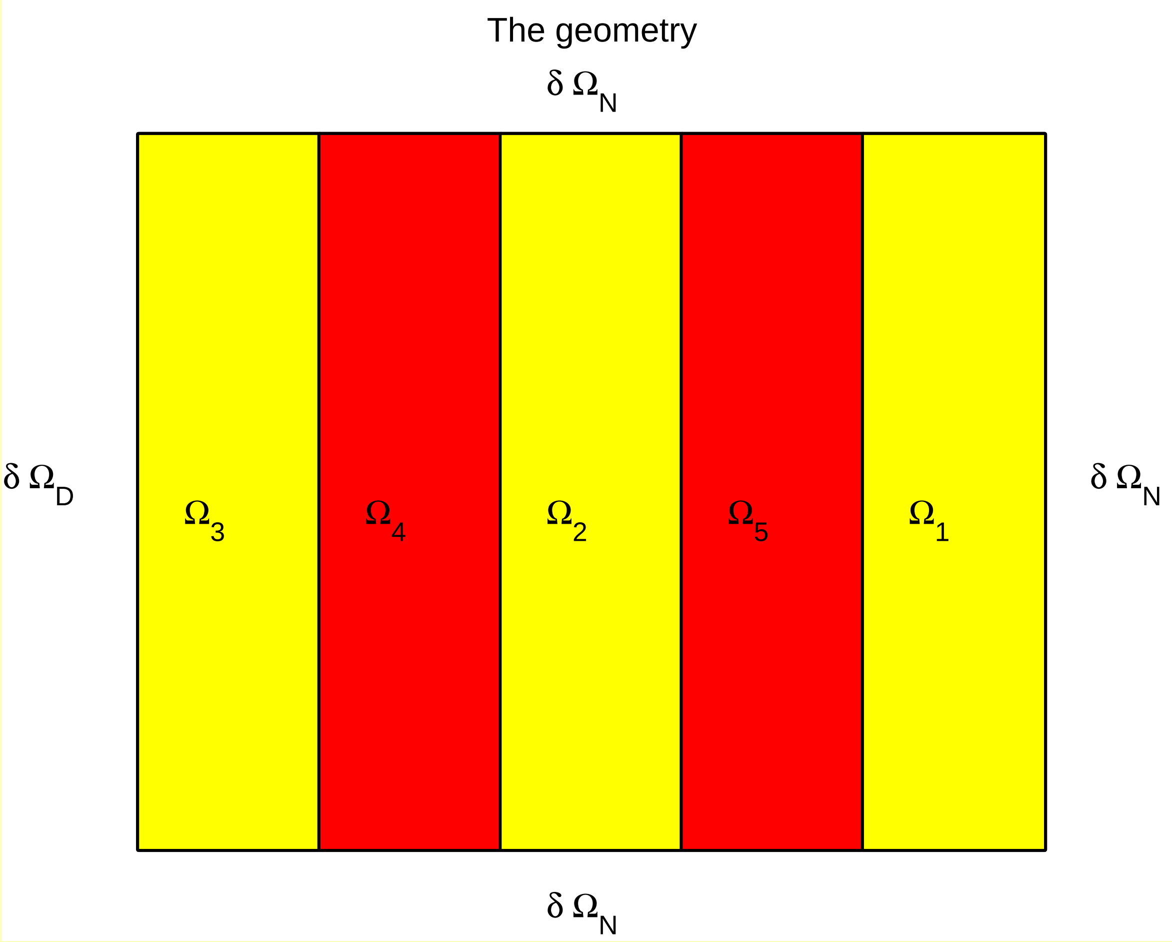
$$-\text{div}(\sigma_j \nabla z_i) = 0 \text{ on } \Omega_j, j \in \{k^h + 1, ..., k\},$$

  with appropriate boundary conditions

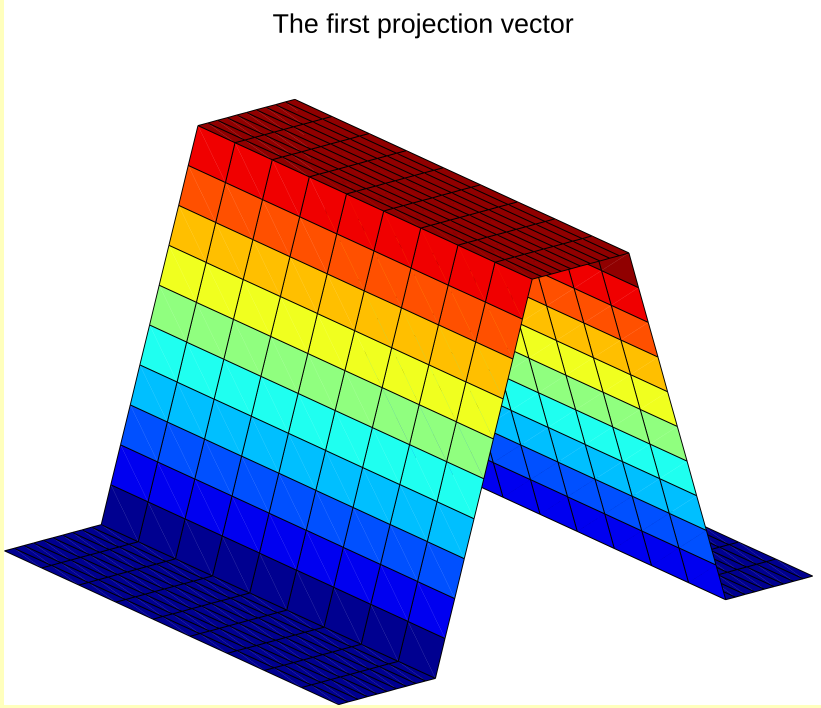
Sparse vectors, Subproblems are cheap to solve

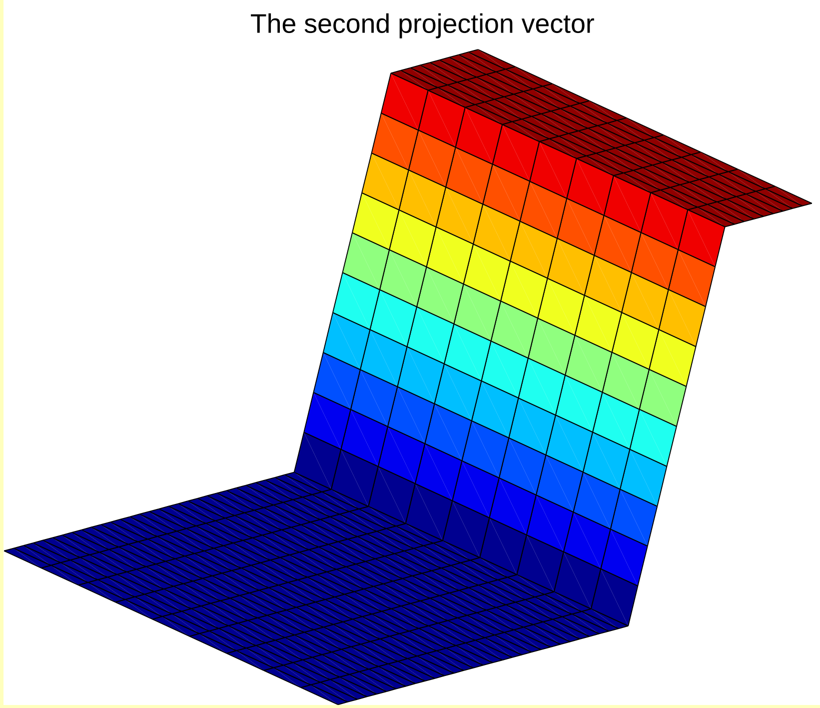
TU Delft

Physical deflation vectors



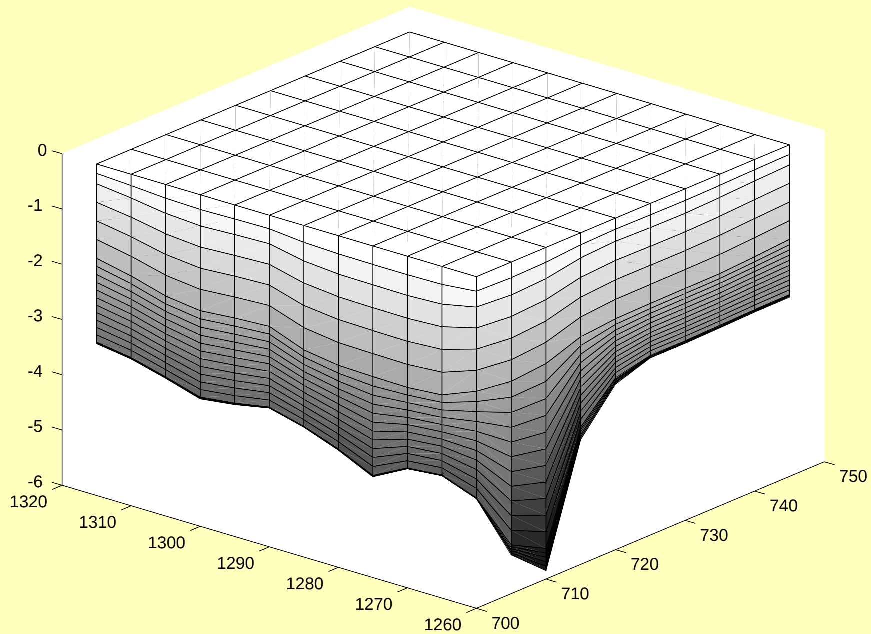
Physical deflation vectors

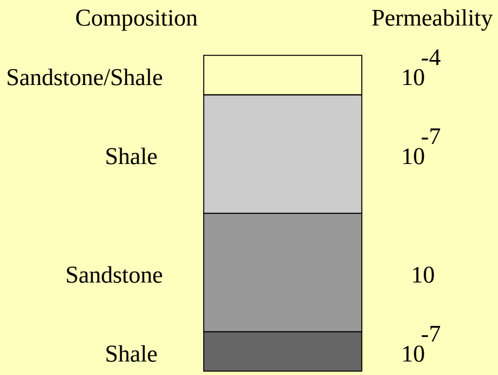




TU Delft

Geometry oil flow problem



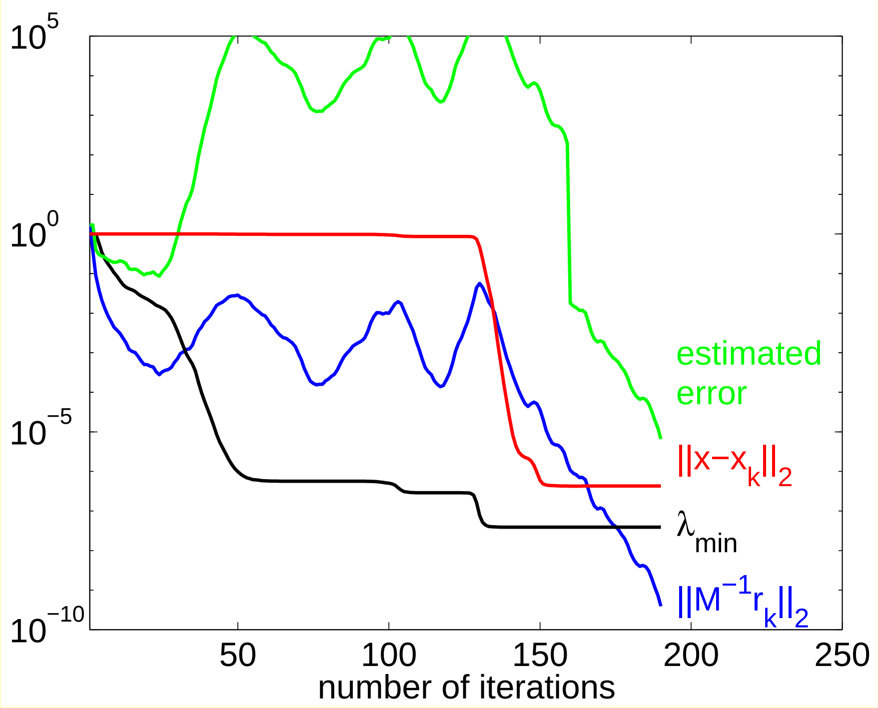


A three-dimensional test problem

TU Delft

Results oil flow problem

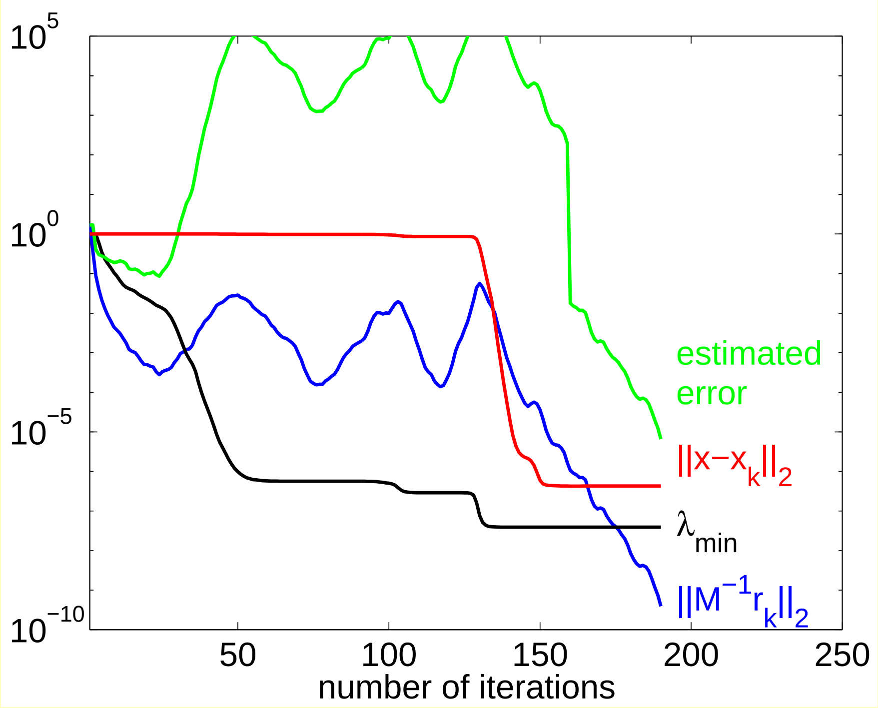
The convergence behavior for ICCG

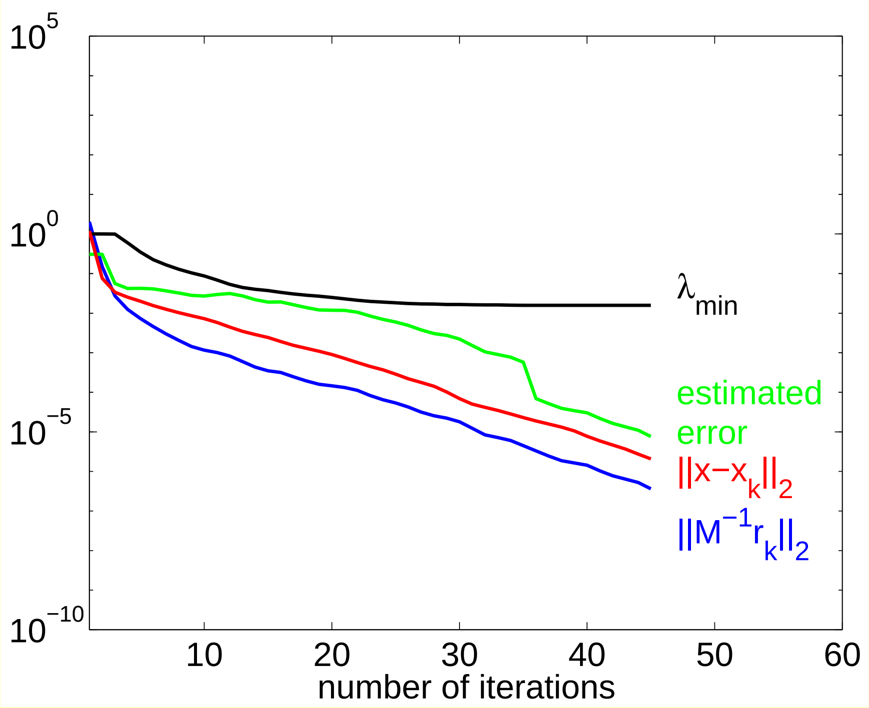


TU Delft

Results oil flow problem

The convergence behavior for ICCG and DICCG





TU Delft

Subdomain deflation

k is number of subdomains

$z_1, ..., z_k$ deflation vectors

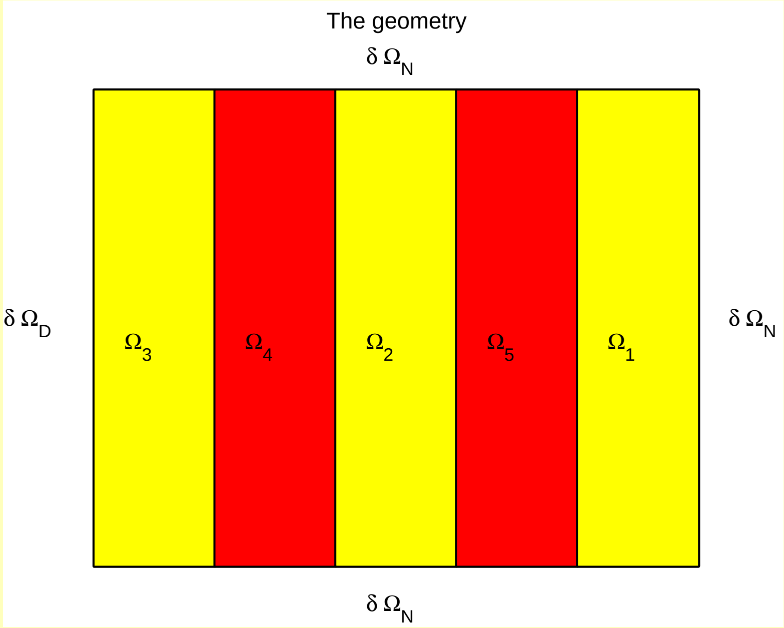
- $z_i = 1$ on $\bar{\Omega}_i$
- $z_i = 0$ on $\Omega \setminus \bar{\Omega}_i$

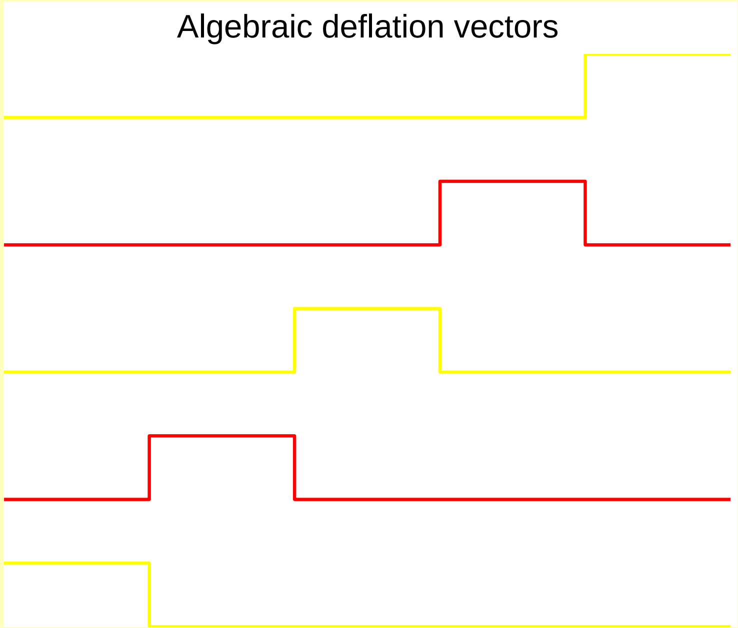
Remarks

- The matrix E is sparse
- $K_{eff}(PA)$ decreases for increasing k
- Work to invert E increases for increasing k
- Optimal value of k?

TU Delft

Algebraic deflation vectors

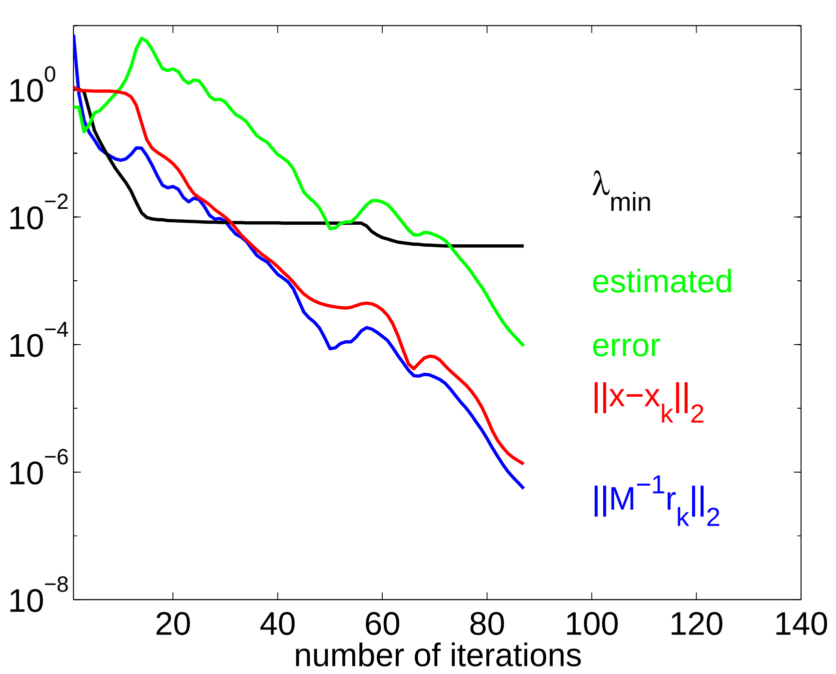




TU Delft

Results (discontinuous coefficients)

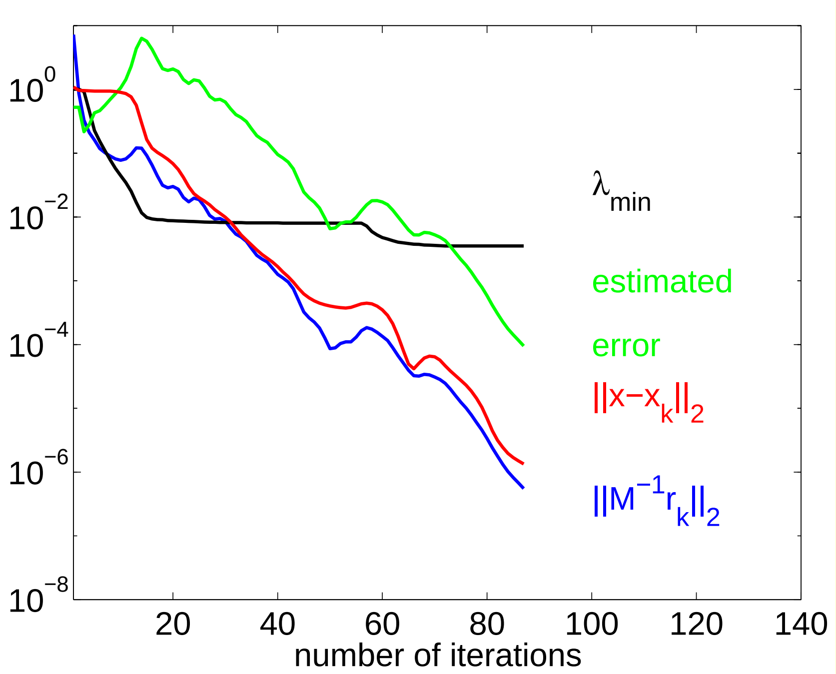
Physical deflation vectors



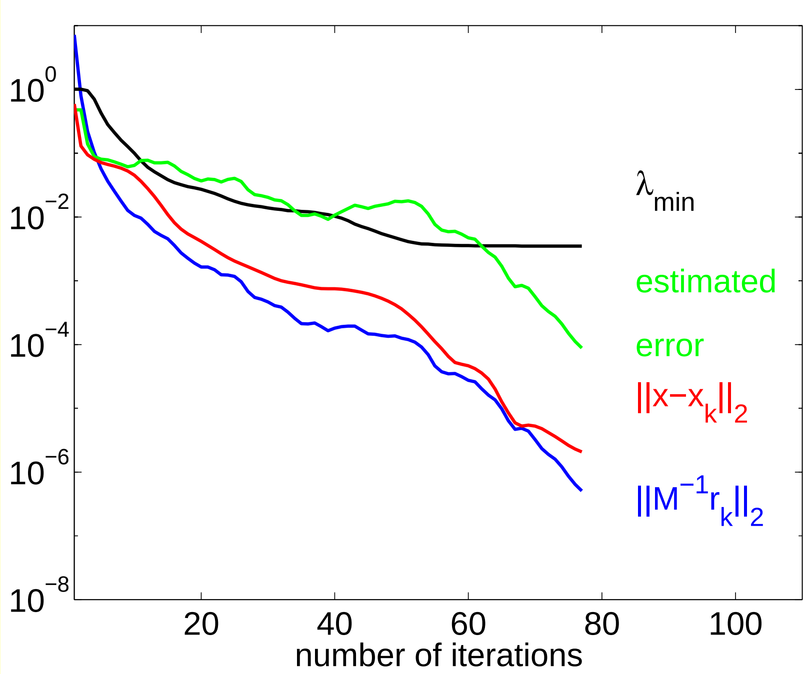
TU Delft

Results (discontinuous coefficients)

Physical deflation vectors



Algebraic deflation vectors



TU Delft

4. Parallelle methoden

Fixed grid 300×300

Number of iterations

p	1	4	25
GMR6	1 (160)	80 (69)	168 (26)
GMR2	3 (50)	86 (16)	192 (11)
GMR1	6 (24)	139 (14)	303 (6)
RILU	160 (1)	341 (1)	437 (1)

Fixed grid 300×300

Number of iterations

p	1	4	25
GMR6	1 (160)	80 (69)	168 (26)
GMR2	3 (50)	86 (16)	192 (11)
GMR1	6 (24)	139 (14)	303 (6)
RILU	160 (1)	341 (1)	437 (1)

Wall clock time Cray T3E

1	4	25
81	685	79
75	167	37
70	222	39
119	65	15

Parallelization of DICCG

Compute and store the sparse vectors

$$c_j = Az_j$$

Compute $E^{-1} = (Z^T AZ)^{-1}$ and store it on each processor
To compute PAv:

1. $w = Av$

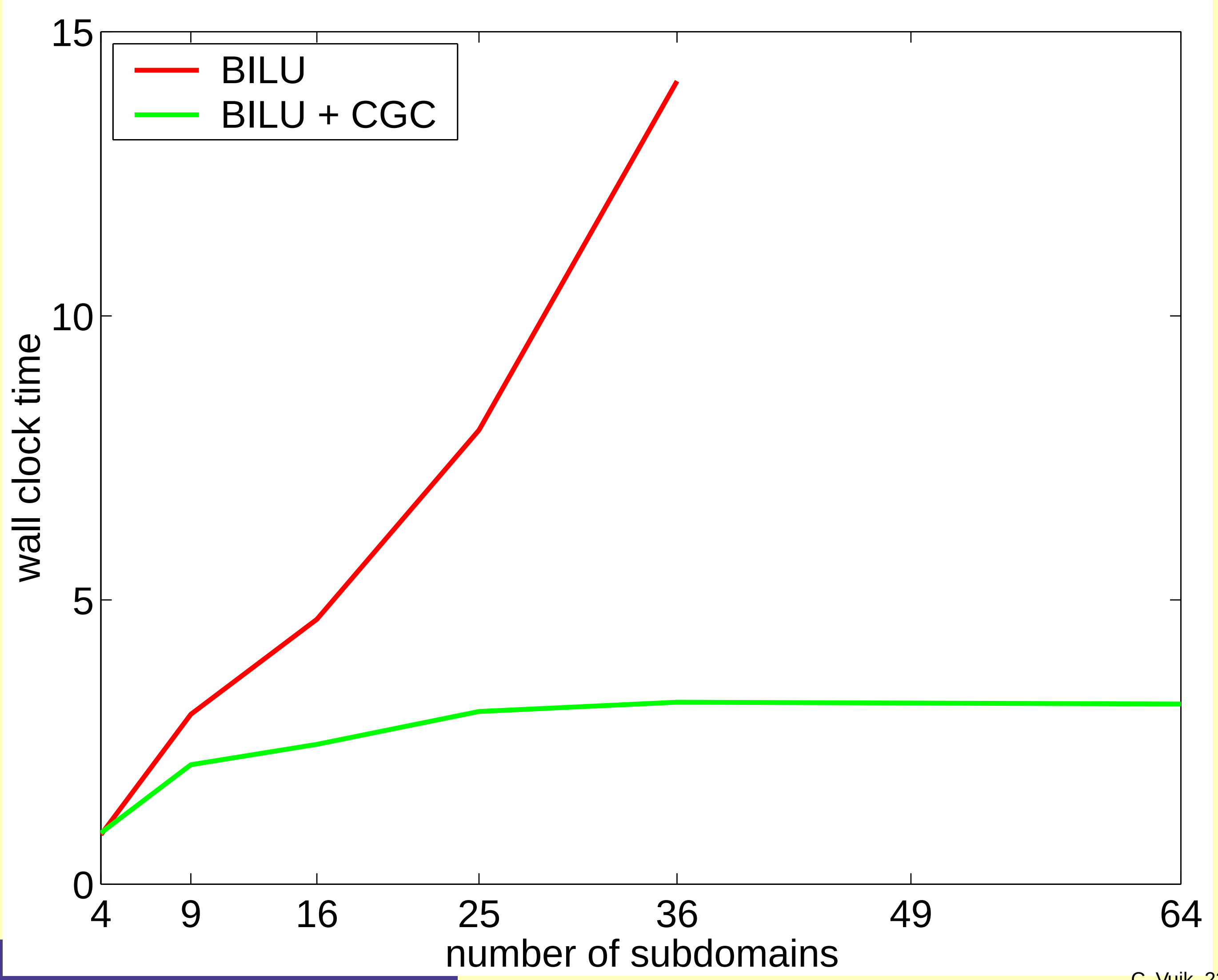
2. Compute the inner products $\tilde{w} = Z^T w$

3. $\tilde{e} = (Z^T AZ)^{-1}\tilde{w}$ on each processor

4. form $v - [c_1 ... c_m]\tilde{e}$

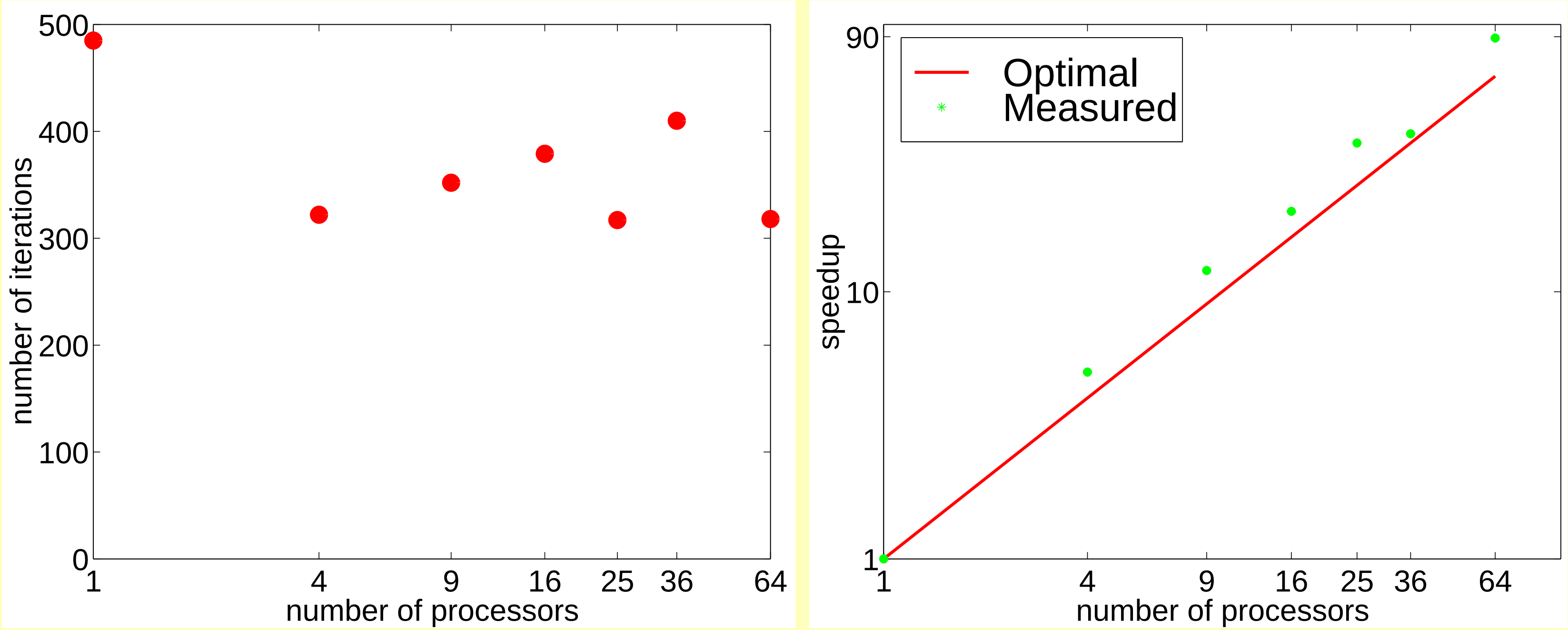
TU Delft

Parallel scalability, subdomain 50×50*, Cray T3E*



TU Delft

Parallel speedup, 480×480 grid, Cray T3E



TU Delft