TU Delft

# Deflation acceleration of block ILU preconditioned Krylov methods

C. Vuik[1] R. Nabben[2] J.M. Tang[1]

[1]Delft University of Technology
Delft Institute of Applied Mathematics
J.M. Burgerscentrum

[2]Technical University Berlin
Institut für Mathematik

IMACS International Symposium on Iterative Methods in Scientific Computing March 17-20, 2008 Lille, France

Delft Institute of Applied Mathematics

# Outline



Delft Institute of Applied Mathematics

# Incompressible Navier-Stokes problems

Discretized incompressible Navier-Stokes

- Momentum equations
- Pressure equation
- Transport equation

Coupled problem

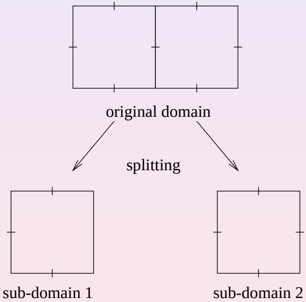
$$\begin{pmatrix} \mathbf{Q} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \; u \in \mathbb{R}^n \text{ and } p \in \mathbb{R}^m$$

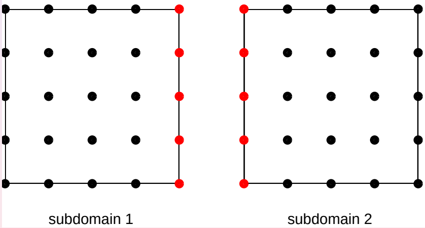
Solve the system $Ax = b$

# Literature review

- Robust preconditioners
  (M)ICCG vd Vorst, Meijering, Gustafsson
  ILUT Saad, MRILU Ploeg, Wubs
  Navier-Stokes Elman, Silvester, Wathen, Golub
  RIF Benzi, Tuma
- Parallel preconditioners Block variants see above
  ILU Bastian, Horton, Vuik, Nooyen, Wesseling
  SPAI Grote, Huckle, Benzi, Tuma, Chow, Saad
- Acceleration of parallel preconditioners
  CGC Notay, vd Velde, Benzi, Frommer, Nabben, Szyld, Chan, Mathew, Dryja, Widlund, Padiy, Axelsson, Polman
  Deflation Nicolaides, Mansfield, Kolotilina, Frank, Vuik Morgan, Chapman, Saad, Burrage, Ehrel, Pohl
  FETI Farhat, Roux, Mandel, Klawonn, Widlund

# Data distribution



# Parallelization of ICCG

**ICCG**

$k = 0,\ r_0 = b - Ax_0,\ p_1 = z_1 = L^{-T}L^{-1}r_0;$

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

$k = k + 1;$

$\alpha_k = \frac{(r_{k-1}, z_{k-1})}{(p_k, Ap_k)};$

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

$r_k = r_{k-1} - \alpha_k A p_k;$

$z_k = L^{-T}L^{-1}r_k;$

$\beta_k = \frac{(r_k, z_k)}{(r_{k-1}, z_{k-1})};$

$p_{k+1} = z_k + \beta_k p_k;$

**end while**

# Explanation for a 1D example

## Building blocks

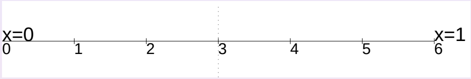
- vector update
- inner product
- matrix vector product
- preconditioner vector product

$$-\frac{d^2y}{dx^2} = f, \quad y(0) = y(1) = 0.$$

Take $n = 5$ and decompose the domain into two subdomains (1 and 2)

TUDelft

# Vector update




We define $I_1 = \{1, 2, 3, \}$ and $I_2 = \{3, 4, 5\}$. Note that there is an overlap of 1 point.

Global vector $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$, local vectors $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ and $\begin{pmatrix} x_3 \\ x_4 \\ x_5 \end{pmatrix}$.

Vector update is straight forward.

# Inner product

- Determine the local innerproduct
- Sum the local innerproducts by MPI_ALLREDUCE

But

The contributions of the interface points are used more than once.

Solution: use the interface points only in one local inner product.

Delft Institute of Applied Mathematics

TUDelft

# Matrix vector product

$$A = \begin{pmatrix} A_{11} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & A_{22} \end{pmatrix}$$

The global matrix vector product $\mathbf{p} = A\mathbf{x}$:

1. Determine $\begin{pmatrix} p_1 \\ p_2 \\ p_3^l \end{pmatrix} = A_{11} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ and $\begin{pmatrix} p_3^r \\ p_4 \\ p_5 \end{pmatrix} = A_{22} \begin{pmatrix} x_3 \\ x_4 \\ x_5 \end{pmatrix}$ in parallel.
2. Communication: send $p_3^l$ from CPU1 to CPU2 and send $p_3^r$ from CPU2 to CPU1. (nearest neighbour communication)
3. Determine on both processors $p_3 = p_3^l + p_3^r$ in parallel.

# Parallelization of a block preconditioner

Take as preconditioner the following

$$\mathbf{p} = P^{-1}\mathbf{x} = \left( \sum_{i=1}^{p} R_i^T P_{i,i}^{-1} R_i \right) \mathbf{x}$$

where

$$P_{i,i} \approx A_{i,i}$$

In our example

$$R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \text{ and } R_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

TUDelft

# Parallelization of a block preconditioner

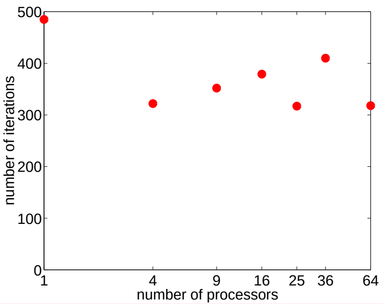
The global preconditioner vector product $\mathbf{p} = P^{-1}\mathbf{x}$:

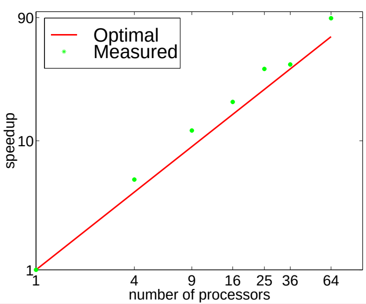
1. Determine $\begin{pmatrix} p_1 \\ p_2 \\ p_3^l \end{pmatrix} = P_{11}^{-1} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ and $\begin{pmatrix} p_3^r \\ p_4 \\ p_5 \end{pmatrix} = P_{22}^{-1} \begin{pmatrix} x_3 \\ x_4 \\ x_5 \end{pmatrix}$ in parallel.
2. Communication: send $p_3^l$ from CPU1 to CPU2 and send $p_3^r$ from CPU2 to CPU1. (nearest neighbour communication)
3. Determine on both processors $p_3 = p_3^l + p_3^r$ in parallel.

Delft Institute of Applied Mathematics

# Parallel results

480 × 480 grid, Cray T3E, DICCG



# 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 | $\sum_{i=1}^{m}(M_i^{-1})PAr$ |

TUDelft

# Deflation operator

$A$ 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. $AP^T = PA$

Delft Institute of Applied Mathematics

TU Delft

# Deflated ICCG

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

$$(I - P^T)x = ZE^{-1}Z^T b, \qquad AP^T x = PAx = Pb.$$

**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**

TUDelft

# Variants for values at interfaces

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

1. **no overlap**

   $z_i = 1$ at one subdomain
   $z_i = 0$ at other subdomains

2. **complete overlap**

   $z_i = 1$ at all subdomains

3. **average overlap**

   $z_i = \frac{1}{n_{neighbors}}$ at all subdomains

4. **weighted overlap** $(-\mathrm{div}(\sigma \nabla u) = f)$

   $$z_i = \frac{\sigma(i)}{\sum \sigma(neighbors)}$$

Delft Institute of Applied Mathematics

TU Delft

# Parallel implementation (initialization)

| Processor 1 | | Processor 2 |
|---|---|---|
| Make $z_1$ | | Make $z_2$ |
| | communication | |
| $z_{2\Gamma}$ | | $z_{1\Gamma}$ |
| Make $Az_1$ and $Az_{2\Gamma}$ | | Make $Az_2$ and $Az_{1\Gamma}$ |
| | communication | |
| sum up | | sum up |
| $E_{11} = z_1^T A z_1$, $E_{12} = z_1^T A z_{2\Gamma}$ | | $E_{22} = z_2^T A z_2$, $E_{12} = z_2^T A z_{1\Gamma}$ |
| | communication | |
| Determine Choleski decomposition of $E$ | | |

Delft Institute of Applied Mathematics

TU Delft

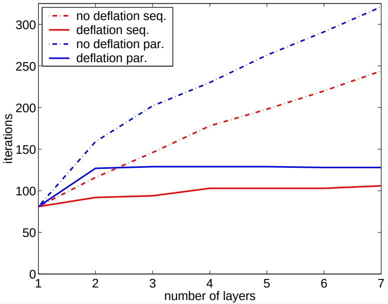
# Parallel implementation (during iteration)

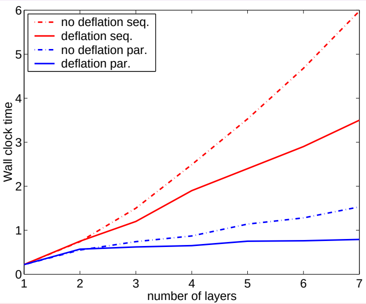
$$P\mathbf{v} = \mathbf{v} - AZ(Z^TAZ)^{-1}Z^T\mathbf{v} = \mathbf{v} - AZE^{-1}Z^T\mathbf{v}$$

| Processor 1 | | Processor 2 |
|---|---|---|
| Compute $z_1^T v$ | | Compute $z_2^T v$ |
| | communication | |
| $y = E^{-1}\begin{pmatrix} z_1^T v \\ z_2^T v \end{pmatrix}$ | | |
| | communication | |
| $\mathbf{v} - y_1 A z_1 - y_2 A z_{2\Gamma}$ | | $\mathbf{v} - y_1 A z_{1\Gamma} - y_2 A z_2$ |

# Numerical results

## Poisson on parallel layers




Delft Institute of Applied Mathematics

# Idea of second-level preconditioners

Various choices are possible:

- Projection vectors
  Physical vectors, eigenvectors, coarse grid projection vectors (constant, linear, ...)
- Projection method
  Deflation, coarse grid projection, balancing, augmented, FETI, multi-grid, ...
- Implementation
  sparseness, with(out) using projection properties, optimized, robustness, ...

# Literature

## Deflated CG and coarse grid projection vectors

Nicolaides 1987, Mansfield 1990, Graham and Hagger 1997, 1999, 1999, Saad, Yeung, Erhel and Guyomarc'h 2000, Frank and Vuik 2001, Rodriguez, Klie and Wheeler 2006, Nabben and Vuik 2004, 2006, Tang, Nabben, Vuik and Erlangga 2007, B. Carpentieri, L. Giraud, and S. Gratton, 2007

## Additive Coarse Grid Correction

Bramble, Pasciak and Schatz 1986, Dryja and Widlund 1991, Smith, Bjorstad and Gropp 1996, Benzi, Frommer, Nabben and Szyld 2001, Toselli and Widlund 2005

## Balancing (Neumann-Neumann) preconditioner

Mandel 1993, Dryja and Widlund 1995, Mandel and Brezina 1996, Pavarino and Widlund 2002, Toselli and Widlund 2005

# Comparison of Deflation and Additive Coarse Grid Correction

TU Delft

$$P_D = I - AZE^{-1}Z^T \qquad P_C = I + \sigma ZE^{-1}Z^T$$

$$M^{-1}P_D = M^{-1} - M^{-1}AZE^{-1}Z^T \qquad P_{CM^{-1}} = M^{-1} + \sigma ZE^{-1}Z^T$$

where $E = Z^TAZ$.

Work per iteration:

- 1 matrix vector product
- 1 preconditioner vector product
- 1 coarse grid operator

# Comparison for eigenvectors

**Definition**

Eigenpair $\{\lambda_i, v_i\}$, so $Av_i = \lambda_i v_i$ with $0 < \lambda_1 \leq \ldots \leq \lambda_n$.
Take $Z = [v_1 \ldots v_r]$.

**Theorem**

- the spectrum of $P_D A$ is $\{0, \ldots, 0, \lambda_{r+1}, \ldots, \lambda_n\}$
- the spectrum of $P_C A$ is $\{\sigma + \lambda_1, \ldots, \sigma + \lambda_r, \lambda_{r+1}, \ldots, \lambda_n\}$

# Comparison for eigenvectors

## Corollary

$$cond_{eff}(P_D A) = \frac{\lambda_n}{\lambda_{r+1}} \leq \frac{\max\{\lambda_n, \sigma + \lambda_r\}}{\min\{\lambda_{r+1}, \sigma + \lambda_1\}} = cond(P_C A)$$

- The eigenvalues of $P_C A$ has a worse distribution than the eigenvalues of $P_D A$

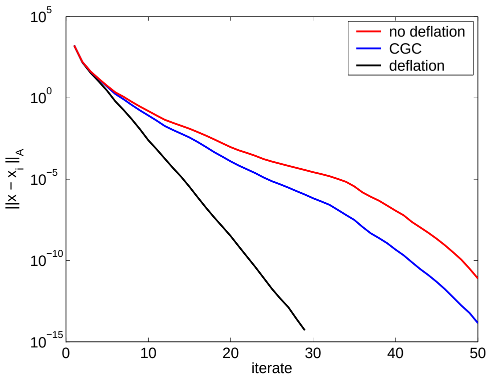
## Conclusion

Deflation is asymptotically better than additive coarse grid correction!

TU Delft

# Results for eigenvectors

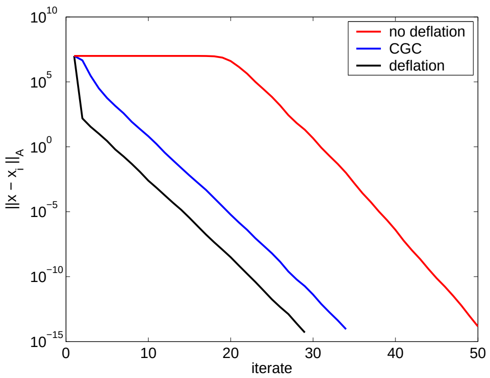
The eigenvalues of $A$ are $1, 2, 3, \ldots, 99, 100$. The eigenvectors $v_1, \ldots, v_{10}$ are used as projection vectors.




Delft Institute of Applied Mathematics

# Results for eigenvectors

The eigenvalues of $A$ are $10^{-6}, \ldots 10^{-6}, 11, 12, 13, \ldots, 99, 100$.
The eigenvectors $v_1, \ldots, v_{10}$ are used as projection vectors.



TUDelft

# Comparison for general projection vectors

**Definition**

$$P_{CM^{-1}} := M^{-1} + \sigma Z E^{-1} Z^T.$$

**Theorem**

Let $A$ and $M$ be symmetric positive definite. Let $Z \in \mathbb{R}^{n \times r}$ with $rank Z = r$. Let $E := Z^T A Z$. Then

$$\begin{aligned} \lambda_n(M^{-1}P_D A) &\leq \lambda_n(P_{CM^{-1}}A), \\ \lambda_{r+1}(M^{-1}P_D A) &\geq \lambda_1(P_{CM^{-1}}A). \end{aligned}$$

**Corollary**

DICCG converges faster than CICCG for general projection vectors.

Delft Institute of Applied Mathematics

# Comparison of Deflation and the Balancing preconditioner

TU Delft

$$M^{-1}P_D = M^{-1} - M^{-1}AZE^{-1}Z^T$$
$$P_B = (I - ZE^{-1}Z^TA)M^{-1}(I - AZE^{-1}Z^T) + ZE^{-1}Z^T$$
$$P_B = P_D^TM^{-1}P_D + ZE^{-1}Z^T$$

Work per iteration:

| | Deflation | Balancing (depends on implementation) |
|---|---|---|
| matrix vector product | 1 | 3 |
| precon vector product | 1 | 1 |
| coarse grid operator | 1 | 2 |

# Comparison for general vectors

Take $Z = [v_1 \dots v_r]$ and $M = I$.

Theorem

- the spectrum of $P_D A$ is $\{0, \dots, 0, \lambda_{r+1}, \dots, \lambda_n\}$

- the spectrum of $P_B A$ is $\{1, \dots, 1, \lambda_{r+1}, \dots, \lambda_n\}$

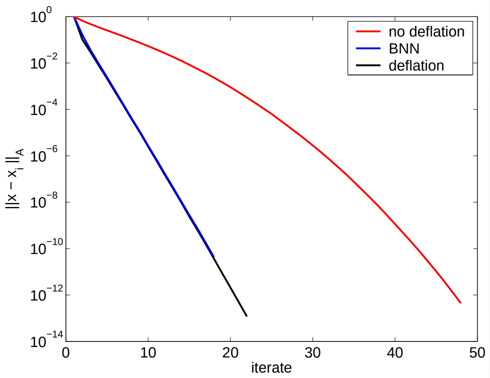
$$cond_{eff}(P_D A) = \frac{\lambda_n}{\lambda_{r+1}} \leq \frac{\max\{\lambda_n, 1\}}{\min\{\lambda_{r+1}, 1\}} = cond(P_B A)$$

Deflation is asymptotically better than Balancing!

TUDelft

# Results for eigenvectors $v_1, \ldots, v_{10}$

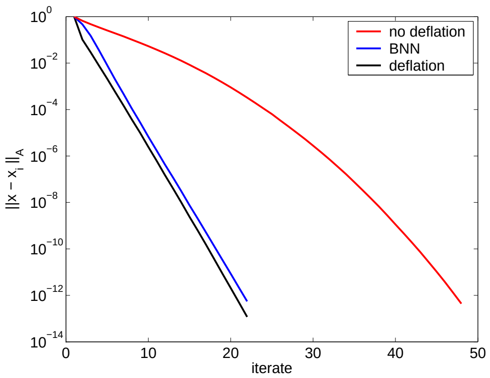
The eigenvalues of $A$ are $0.01, 0.02, 0.03, \ldots, 0.99, 1$.




Delft Institute of Applied Mathematics

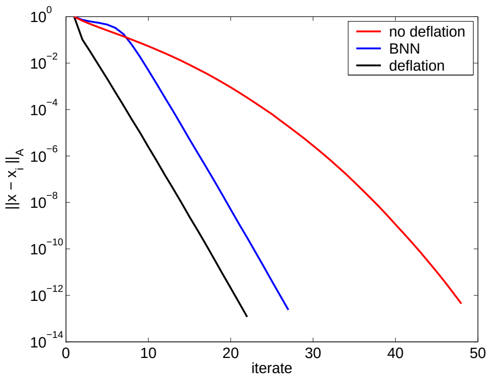
# Results for eigenvectors $v_1, \ldots, v_{10}$

The eigenvalues of $A$ are $0.1, 0.2, 0.3, \ldots, 9.9, 10$.



# Results for eigenvectors $v_1, \ldots, v_{10}$

The eigenvalues of $A$ are $1, 2, 3, \ldots, 99, 100$.



TUDelft

# Conclusions

- Block preconditioned Krylov methods combined with Deflation, additive coarse grid correction, or Balancing are well parallelizable (scalable, good speed up).
- The choice of the projection vectors is important for the success of a projection method.
- Deflation needs less iterations than additive coarse grid correction, and uses the same amount of work per iteration
- Deflation uses less (approximately the same) iterations as Balancing, but uses less work per iteration.

# Further information

- http://ta.twi.tudelft.nl/nw/users/vuik/pub_it_def.html


- 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 J. on Numerical Analysis, 42, pp. 1631-1647, 2004
  SIAM Journal on Scientific Computing, 27, pp. 1742-1759, 2006
- J.M. Tang, R. Nabben, C. Vuik, and Y.A. Erlangga
  Theoretical and numerical comparison of various projection methods derived from deflation, domain decomposition and multigrid methods
  Delft University of Technology Report 07-04