Fast and robust preconditioners for the incompressible Navier-Stokes equations

C. Vuik, M. ur Rehman, A. Segal, C.M. Klaij, and X. He
Delft University of Technology
Delft Institute of Applied Mathematics, Delft,
and MARIN, Wageningen, The Netherlands.

Fifth International Conference on Numerical Algebra and Scientific Computing
Shanghai, P.R. China
October 27, 2014
Messages

1. Incompressible Navier-Stokes are important

2. Much progress in solvers for academic test problems

3. Transfer methods to industrial problems
Outline

1. Introduction
2. Problem
3. Krylov solvers and preconditioners
4. ILU-type preconditioners
5. Block preconditioners
   - SIMPLE
   - Augmented Lagrangian
6. Maritime Applications
7. Conclusions
1. Introduction

Flow in arteries
Introduction

Flooding of the Netherlands, 1953
Introduction

Streamlines around the stern and the axial velocity field in the wake.
2. Problem

\[-\nu \nabla^2 u + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p = f \quad \text{in} \quad \Omega\]

\[\nabla \cdot \mathbf{u} = 0 \quad \text{in} \quad \Omega.\]

\(\mathbf{u}\) is the fluid velocity vector
\(p\) is the pressure field
\(\nu > 0\) is the kinematic viscosity coefficient \((1/\text{Re})\).
\(\Omega \subset \mathbb{R}^2 \text{ or }^3\) is a bounded domain with the boundary condition:

\[\mathbf{u} = \mathbf{w} \quad \text{on } \partial \Omega_D, \quad \nu \frac{\partial \mathbf{u}}{\partial \mathbf{n}} - \mathbf{n} p = 0 \quad \text{on } \partial \Omega_N.\]
Linear system

Matrix form after linearization and discretization:

\[
\begin{bmatrix}
F & B^T \\
B & 0
\end{bmatrix}
\begin{bmatrix}
u \\
p
\end{bmatrix}
=
\begin{bmatrix}
f \\
g
\end{bmatrix}
\]

where \( F \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{m \times n}, f \in \mathbb{R}^n \) and \( m \leq n \)

- \( F = \nu A \) in Stokes problem, \( A \) is vector Laplacian matrix
- \( F = \nu A + N \) in Picard linearization, \( N \) is vector-convection matrix
- \( F = \nu A + N + W \) in Newton linearization, \( W \) is the Newton derivative matrix
- \( B \) is the divergence matrix
- Sparse linear system, Symmetric indefinite (Stokes problem), nonsymmetric otherwise.
- Saddle point problem having large number of zeros on the main diagonal
3. Krylov Solvers and preconditioners

• **Direct method:**
  To solve $Ax = b$,
  factorize $A$ into upper $U$ and lower $L$ triangular matrices ($LUx = b$)
  First solve $Ly = b$, then $Ux = y$

• **Classical Iterative Schemes:**
  Methods based on matrix splitting, generates sequence of iterations
  
  $x_{k+1} = M^{-1}(Nx_k + b) = Qx_k + s$, where $A = M - N$
  
  Jacobi, Gauss Seidel, SOR, SSOR

• **Krylov Subspace Methods:**
  
  $x_{k+1} = x_k + \alpha_k p_k$
  
  Some well known methods are
  
  CGNR[1975], QMR[1991], CGS[1989], Bi-CGSTAB[1992], GMRES[1986],
  GMRESR[1994], GCR[1986], IDR(s)[2007]
IDR and IDR($s$) (Induced Dimension Reduction)

- Sonneveld developed IDR in the 1970’s. IDR is a finite termination (Krylov) method for solving nonsymmetric linear systems.

- Analysis showed that IDR can be viewed as Bi-CG combined with linear minimal residual steps.

- This discovery led to the development of first CGS, and later of Bi-CGSTAB (by van der Vorst).
IDR and IDR($s$) (continued)

• As a result of these developments the basic IDR-idea was abandoned for the Bi-CG-approach.

• Recently, Sonneveld and van Gijzen discovered that the IDR-approach was abandoned too soon and proposed a generalization of IDR: IDR($s$).

• **P. Sonneveld and M.B. van Gijzen** IDR($s$): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations
  

More information: http://ta.twi.tudelft.nl/nw/users/gijzen/IDR.html
4. ILU-type Preconditioners

A linear system $Ax = b$ is transformed into $P^{-1}Ax = P^{-1}b$ such that

- $P \approx A$
- Eigenvalues of $P^{-1}A$ are more clustered than $A$
- $Pz = r$ cheap to compute

Several approaches, we will discuss here

- ILU preconditioner
- Preconditioned IDR($s$) and Bi-CGSTAB comparison
- Block preconditioners
SILU preconditioners

New renumbering Scheme

- Renumbering of grid points:
  - Sloan algorithm [Sloan - 1986]
  - Cuthill McKee algorithms [Cuthill McKee - 1969]
- The unknowns are reordered by p-last or p-last per level methods
  - In **p-last reordering**, first all the velocity unknowns are ordered followed by pressure unknowns. Usually it produces a large profile but avoids breakdown of \( LU \) decomposition.
  - In **p-last per level reordering**, unknowns are reordered per level such that at each level, the velocity unknowns are followed by the pressure unknowns.

what are the levels?
SILU preconditioner

$4 \times 4$ Q2-Q1 grid
The iteration is stopped if the linear systems satisfy $\frac{\|r^k\|_2}{\|b\|_2} \leq tol$. 

\[
\begin{align*}
\text{Driven cavity flow problem} \\
\text{Backward facing step problem}
\end{align*}
\]
Numerical experiments (SILU preconditioners)

Stokes Problem in a square domain with Bi-CGSTAB, 
\( \text{accuracy} = 10^{-6} \), Sloan renumbering

<table>
<thead>
<tr>
<th>Grid size</th>
<th>( Q2 - Q1 )</th>
<th>( Q2 - P1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p-last</td>
<td>p-last per level</td>
</tr>
<tr>
<td>16 × 16</td>
<td>36(0.11)</td>
<td>25(0.09)</td>
</tr>
<tr>
<td>32 × 32</td>
<td>90(0.92)</td>
<td>59(0.66)</td>
</tr>
<tr>
<td>64 × 64</td>
<td>255(11.9)</td>
<td>135(6.7)</td>
</tr>
<tr>
<td>128 × 128</td>
<td>472(96)</td>
<td>249(52)</td>
</tr>
</tbody>
</table>
Numerical Experiments (IDR(s) vs Bi-CGSTAB)

SILU preconditioned: Comparison of iterative methods for increasing stretch factor for the driven cavity Stokes problem.

![Graph showing comparison of iterative methods for increasing stretch factor for the driven cavity Stokes problem.](image-url)
Numerical Experiments (IDR(s) vs Bi-CGSTAB(l))

SILU preconditioned: Comparison of iterative methods

**Driven Cavity Stokes problem, stretch factor 10**

<table>
<thead>
<tr>
<th>Grid</th>
<th>Bi-CGSTAB(l)</th>
<th>IDR(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mat.-Vec.(ts)</td>
<td>l Mat.-Vec.(ts)</td>
</tr>
<tr>
<td>128 × 128</td>
<td>1104(36.5)</td>
<td>4 638(24.7) 6</td>
</tr>
<tr>
<td>256 × 256</td>
<td>5904(810)</td>
<td>6 1749(307) 8</td>
</tr>
</tbody>
</table>

**Channel flow Stokes problem, length 100**

<table>
<thead>
<tr>
<th>Grid</th>
<th>Bi-CGSTAB(l)</th>
<th>IDR(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mat.-Vec.(ts)</td>
<td>l Mat.-Vec.(ts)</td>
</tr>
<tr>
<td>64 × 64</td>
<td>1520(12)</td>
<td>4 938(8.7) 8</td>
</tr>
<tr>
<td>128 × 128</td>
<td>NC</td>
<td>6 8224(335) 8</td>
</tr>
</tbody>
</table>
5. Block preconditioners

\[ A = L_b D_b U_b = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ BM_l^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & M_u^{-1}B^T \\ 0 & I \end{bmatrix} \]

\[ M_l = M_u = F \text{ and } S = -BF^{-1}B^T \text{ is the Schur-complement matrix.} \]

\[ \mathcal{U}_{bt} = D_b U_b = \begin{bmatrix} F & B^T \\ 0 & \hat{S} \end{bmatrix}, \quad \mathcal{L}_{bt} = L_b D_b = \begin{bmatrix} F & 0 \\ B & \hat{S} \end{bmatrix}. \]

Preconditioners are based on combination of these blocks involve:

\[ Fz_1 = r_1 \text{ The velocity subsystem} \]

\[ S \rightarrow \hat{S} \]

\[ \hat{S}z_2 = r_2 \text{ The pressure subsystem} \]
Block preconditioners

Block triangular preconditioners

\[ P_t = U_{bt} = \begin{bmatrix} F & B^T \\ 0 & \hat{S} \end{bmatrix} \]

- Pressure convection diffusion (PCD) [Kay et al, 2002]
  \[ \hat{S} = -A_p F_p^{-1} Q_p, \ Q_p \text{ is the pressure mass matrix} \]

- Least squares commutator (LSC) [Elman et al, 2002]
  \[ \hat{S} = -(BQ_u^{-1} B^T)(BQ_u^{-1} FQ_u^{-1} B^T)^{-1}(BQ_u^{-1} B^T), \ Q_u \text{ is the velocity mass matrix} \]

- Augmented Lagrangian approach (AL) [Benzi and Olshanskii, 2006]
  \[ F \text{ is replaced by } F_{\gamma} = F + \gamma B W^{-1} B^T \]
  \[ \hat{S}^{-1} = -\left(\nu \hat{Q}_p^{-1} + \gamma W^{-1}\right), \ W = \hat{Q}_p \]
Block preconditioners (SIMPLE)

**SIMPLE-type preconditioners**[Vuik et al-2000]

<table>
<thead>
<tr>
<th>SIMPLE</th>
<th>SIMPLER</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z = U_b^{-1} L_{bt}^{-1} r$</td>
<td>$z = U_{bt}^{-1} L_b^{-1} r$</td>
</tr>
<tr>
<td>$M_u = D$</td>
<td>$M_l = M_u = D$, $D = diag(F)$</td>
</tr>
<tr>
<td>$\hat{S} = BD^{-1}B^T$</td>
<td>$\hat{S} = BD^{-1}B^T$</td>
</tr>
<tr>
<td>One Poisson solve</td>
<td>Two Poisson solves</td>
</tr>
<tr>
<td>One velocity solve</td>
<td>Two velocity solves</td>
</tr>
</tbody>
</table>

**Lemma:** In the SIMPLER preconditioner/algorithm, both variants (one or two velocity solves) are identical.
Improvements in SIMPLE-type preconditioners

We use approximate solvers for subsystems, so flexible Krylov solvers are required (GCR, FGMRES, GMRESR)

- hSIMPLER
- MSIMPLER
Improvements in SIMPLE(R) preconditioners

hSIMPLER preconditioner:

In hSIMPLER (hybrid SIMPLER), first iteration of Krylov method preconditioned with SIMPLER is done with SIMPLE and SIMPLER is employed afterwards.

- Faster convergence than SIMPLER
- Effective in the Stokes problem
Improvements in SIMPLE(R) preconditioners

**MSIMPLER preconditioner:**
Making the following changes in SIMPLER leads to the MSIMPLER preconditioner.

LSC: \( \hat{S} \approx -(B\hat{Q}_u^{-1}B^T)(B\hat{Q}_u^{-1}F\hat{Q}_u^{-1}B^T)^{-1}(B\hat{Q}_u^{-1}B^T) \)

assuming \( F\hat{Q}_u^{-1} \approx I \) (time dependent problems with a small time step)

\[ \hat{S} = -B\hat{Q}_u^{-1}B^T \]

MSIMPLER uses this approximation for the Schur complement and updates scaled with \( \hat{Q}_u^{-1} \).

- Convergence better than other variants of SIMPLE
- Cheaper than SIMPLER (in construction) and LSC (per iteration)
Numerical Experiments (comparison)

3D Backward facing step: Preconditioners used in the Stokes problem with preconditioned GCR(20) with accuracy of $10^{-6}$ (SEPRAN) using Q2-Q1 hexahedrons

<table>
<thead>
<tr>
<th>Grid</th>
<th>SIMPLE</th>
<th>LSC</th>
<th>MSIMPLER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>iter. ($t_s$)</td>
<td>in-it-$u$</td>
<td>in-it-$p$</td>
</tr>
<tr>
<td>$8 \times 8 \times 16$</td>
<td>44(4) $\frac{97}{342}$</td>
<td>16(1.9) $\frac{41}{216}$</td>
<td>14(1.4) $\frac{28}{168}$</td>
</tr>
<tr>
<td>$16 \times 16 \times 32$</td>
<td>84(107) $\frac{315}{1982}$</td>
<td>29(51) $\frac{161}{1263}$</td>
<td>17(21) $\frac{52}{766}$</td>
</tr>
<tr>
<td>$24 \times 24 \times 48$</td>
<td>99(447) $\frac{339}{3392}$</td>
<td>26(233) $\frac{193}{2297}$</td>
<td>17(77) $\frac{46}{1116}$</td>
</tr>
<tr>
<td>$32 \times 32 \times 40$</td>
<td>132(972) $\frac{574}{5559}$</td>
<td>37(379) $\frac{233}{2887}$</td>
<td>20(143) $\frac{66}{1604}$</td>
</tr>
</tbody>
</table>
Numerical Experiments (comparison)

3D Lid driven cavity problem (tetrahedrons): The Navier-Stokes problem is solved with accuracy $10^{-4}$, a linear system at each Picard step is solved with accuracy $10^{-2}$ using preconditioned Krylov subspace methods. Bi-CGSTAB is used as inner solver in block preconditioners (SEPRAN)

<table>
<thead>
<tr>
<th>Re</th>
<th>LSC</th>
<th>MSIMPLER</th>
<th>SILU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GCR iter. $(t_s)$</td>
<td>GCR iter. $(t_s)$</td>
<td>Bi-CGSTAB iter. $(t_s)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 × 16 × 16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>30(20)</td>
<td>20(16)</td>
<td>144(22)</td>
</tr>
<tr>
<td>50</td>
<td>57(37)</td>
<td>37(24)</td>
<td>234(35)</td>
</tr>
<tr>
<td>100</td>
<td>120(81)</td>
<td>68(44)</td>
<td>427(62)</td>
</tr>
<tr>
<td>32 × 32 × 32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>38(234)</td>
<td>29(144)</td>
<td>463(353)</td>
</tr>
<tr>
<td>50</td>
<td>87(544)</td>
<td>53(300)</td>
<td>764(585)</td>
</tr>
<tr>
<td>100</td>
<td>210(1440)</td>
<td>104(654)</td>
<td>1449(1116)</td>
</tr>
</tbody>
</table>
Numerical Experiments (comparison)

2D Lid driven cavity problem on $64 \times 64$ stretched grid: The Stokes problem is solved with accuracy $10^{-6}$. PCG is used as inner solver in block preconditioners (SEPRAN).

<table>
<thead>
<tr>
<th>Stretch factor</th>
<th>LSC GCR iter.</th>
<th>MSIMPLER GCR iter.</th>
<th>SILU Bi-CGSTAB iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>17</td>
<td>96</td>
</tr>
<tr>
<td>8</td>
<td>49</td>
<td>28</td>
<td>189</td>
</tr>
<tr>
<td>16</td>
<td>71</td>
<td>34</td>
<td>317</td>
</tr>
<tr>
<td>32</td>
<td>97</td>
<td>45</td>
<td>414</td>
</tr>
<tr>
<td>64</td>
<td>145</td>
<td>56</td>
<td>NC</td>
</tr>
<tr>
<td>128</td>
<td>NC</td>
<td>81</td>
<td>NC</td>
</tr>
</tbody>
</table>
The Augmented Lagrangian method

\[
\begin{bmatrix}
F & B^T \\
B & O
\end{bmatrix}
\begin{bmatrix}
u \\
p
\end{bmatrix} =
\begin{bmatrix}
f \\
g
\end{bmatrix}
\]
is transformed into

\[
\begin{bmatrix}
F + \gamma B^T W^{-1} B & B^T \\
B & 0
\end{bmatrix}
\begin{bmatrix}
u \\
p
\end{bmatrix} =
\begin{bmatrix}
\hat{f} \\
g
\end{bmatrix}
\] or \( \mathcal{A}_{AL} \mathbf{x} = \hat{b} \),

with \( \hat{f} = f + \gamma B^T W^{-1} B g \), where \( W \) is a non-singular matrix.

The Ideal AL preconditioner proposed for \( \mathcal{A}_{AL} \) is

\[
\mathcal{P}_{IAL} =
\begin{bmatrix}
F + \gamma B^T W^{-1} B & 0 \\
B & -\frac{1}{\gamma} W
\end{bmatrix}.
\]
The Augmented Lagrangian method

\[ A_{AL} = \begin{bmatrix} F + \gamma B^T W^{-1} B & B^T \\ B & 0 \end{bmatrix} \quad (S_{AL} = -B(F + \gamma B^T W^{-1} B)^{-1} B^T) \]

\[ P_{IAL} = \begin{bmatrix} F + \gamma B^T W^{-1} B & 0 \\ B & -\frac{1}{\gamma} W \end{bmatrix} \quad (F_\gamma = F + \gamma B^T W^{-1} B) \]

- The Schur complement \( S_{AL} \) of \( A_{AL} \) is approximated by \(-\frac{1}{\gamma} W\).
- The block \( F_\gamma \) becomes increasingly ill-conditioned with \( \gamma \to \infty \).
- In practice it is often chosen as \( \gamma = 1 \), or \( \gamma = O(1) \), and \( W = \hat{Q} P \).
- Open question: fast solution methods for systems with \( F_\gamma \), which is denser than \( F \) and consists of mixed derivatives.

The Augmented Lagrangian method

The transformed coefficient matrix $A_{AL} = \begin{bmatrix} F + \gamma B^T W^{-1} B & B^T \\ B & 0 \end{bmatrix}$ and the ideal AL precondition $P_{IAL} = \begin{bmatrix} F + \gamma B^T W^{-1} B & 0 \\ B & -\frac{1}{\gamma} W \end{bmatrix}$ includes (in 2D)

- the convection-diffusion block: $F = \begin{bmatrix} F_{11} & O \\ O & F_{11} \end{bmatrix}$,
- the (negative) divergence matrix: $B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$,
- the modified pivot block $F_\gamma = \begin{bmatrix} F_{11} + \gamma B_1^T W^{-1} B_1 & \gamma B_1^T W^{-1} B_2 \\ \gamma B_2^T W^{-1} B_1 & F_{11} + \gamma B_2^T W^{-1} B_2 \end{bmatrix}$.

One approximation of $F_\gamma$ is $\tilde{F}_\gamma = \begin{bmatrix} F_{11} + \gamma B_1^T W^{-1} B_1 & O \\ \gamma B_2^T W^{-1} B_1 & F_{11} + \gamma B_2^T W^{-1} B_2 \end{bmatrix}$, which leads to the modified AL preconditioner $P_{MAL}$ for $A_{AL}$. 
The Augmented Lagrangian method

\[ P_{IAL} = \begin{bmatrix} F_\gamma & 0 \\ B & -\frac{1}{\gamma} W \end{bmatrix} \quad (F_\gamma = \begin{bmatrix} F_{11} + \gamma B_1^T W^{-1} B_1 & \gamma B_1^T W^{-1} B_2 \\ \gamma B_2^T W^{-1} B_1 & F_{11} + \gamma B_2^T W^{-1} B_2 \end{bmatrix}) \]

\[ P_{MAL} = \begin{bmatrix} \tilde{F}_\gamma & 0 \\ B & -\frac{1}{\gamma} W \end{bmatrix} \quad (\tilde{F}_\gamma = \begin{bmatrix} F_{11} + \gamma B_1^T W^{-1} B_1 & 0 \\ \gamma B_2^T W^{-1} B_1 & F_{11} + \gamma B_2^T W^{-1} B_2 \end{bmatrix}) \]

- systems with \( \tilde{F}_\gamma \) are easier to be solved, compared to \( F_\gamma \).
- the number of iterations by using the ideal and modified AL preconditioners are both independent of the mesh refinement, and nearly independent of the Reynolds (viscosity) number.
- by using the modified AL preconditioner, there exists an optimal value of \( \gamma \), which minimises the number of Krylov subspace iterations. The optimal \( \gamma \) is problem dependent, but mesh size independent.
Numerical experiments (Lid driven cavity)

2D lid driven cavity problem. the domain is $[0, 1] \times [0, 1]$. The Reynolds number is $Re = UL/\nu$, and here $U = 1$ and $L = 1$. The stretched grids are generated based on the uniform Cartesian grids with $n \times n$ cells. The stretching function is applied in both directions with parameters $a = 1/2$ and $b = 1.1$

$$x = \frac{(b + 2a)c - b + 2a}{(2a + 1)(1 + c)}, \quad c = \left(\frac{b + 1}{b - 1}\right)^{\frac{\bar{x} - a}{1 - a}}, \quad \bar{x} = 0, 1/n, 2/n, ..., 1.$$
## Numerical experiments (Lid driven cavity)

<table>
<thead>
<tr>
<th>Re</th>
<th>100</th>
<th>400</th>
<th>1000</th>
<th>2500*</th>
<th>5000*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>modified AL preconditioner</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picard iterations:</td>
<td>14</td>
<td>27</td>
<td>33</td>
<td>66</td>
<td>286</td>
</tr>
<tr>
<td>GCR iterations:</td>
<td>5</td>
<td>9</td>
<td>11</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>total time:</td>
<td>22.7</td>
<td>65.1</td>
<td>119.6</td>
<td>457.7</td>
<td>2636.3</td>
</tr>
<tr>
<td><strong>modified 'grad-div' preconditioner</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picard iterations:</td>
<td>13</td>
<td>27</td>
<td>31</td>
<td>51</td>
<td>308</td>
</tr>
<tr>
<td>GCR iterations:</td>
<td>7</td>
<td>11</td>
<td>16</td>
<td>28</td>
<td>24</td>
</tr>
<tr>
<td>total time:</td>
<td>10.8</td>
<td>35.8</td>
<td>64.4</td>
<td>159.5</td>
<td>812.5</td>
</tr>
<tr>
<td><strong>ideal SIMPLER preconditioner</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picard iterations:</td>
<td>14</td>
<td>27</td>
<td>31</td>
<td>51</td>
<td>325</td>
</tr>
<tr>
<td>GCR iterations:</td>
<td>40</td>
<td>53</td>
<td>63</td>
<td>92</td>
<td>107</td>
</tr>
<tr>
<td>total time:</td>
<td>81.5</td>
<td>235.2</td>
<td>508.4</td>
<td>929.7</td>
<td>9548.7</td>
</tr>
</tbody>
</table>
## Numerical experiments (Lid driven cavity)

<table>
<thead>
<tr>
<th>Re</th>
<th>100</th>
<th>400</th>
<th>1000</th>
<th>2500*</th>
<th>5000*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>modified AL preconditioner</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newton iterations:</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>GCR iterations:</td>
<td>8</td>
<td>14</td>
<td>21</td>
<td>33</td>
<td>50</td>
</tr>
<tr>
<td>total time:</td>
<td>14.8</td>
<td>26.2</td>
<td>74.6</td>
<td>194.2</td>
<td>277.1</td>
</tr>
</tbody>
</table>

| **modified 'grad-div' preconditioner** |      |      |      |       |       |
| Newton iterations:           | 6    | 7    | 8    | 9     | 9     |
| GCR iterations:              | 10   | 17   | 28   | 53    | 77    |
| total time:                  | 8.5  | 15.7 | 32.7 | 119.1 | 167.9 |

| **modified SIMPLER preconditioner** |      |      |      |       |       |
| Newton iterations:           | 10   | 8*   | 8*   | 11    | 15    |
| GCR iterations:              | 43   | 82   | 84   | 80    | 90    |
| total time:                  | 68.3 | 102.9| 232.8| 203.2 | 561.6 |
6. Maritime Applications

Container vessel (unstructured grid)

RaNS equations

\[ k-\omega \text{ turbulence model} \]

\[ y^+ \approx 1 \]

Model-scale:

\[ Re = 1.3 \cdot 10^7 \]

13.3m cells

max aspect ratio 1 : 1600
Tanker (block-structured grid)

Model-scale:
Re = 4.6 \cdot 10^6
2.0m cells
max aspect ratio 1 : 7000

Full-scale:
Re = 2.0 \cdot 10^9
2.7m cells
max aspect ratio 1 : 930 000
Discretization

Co-located, cell-centered finite volume discretization of the steady Navier-Stokes equations with Picard linearization leads to linear system:

\[
\begin{bmatrix}
Q_1 & 0 & 0 & G_1 \\
0 & Q_2 & 0 & G_2 \\
0 & 0 & Q_3 & G_3 \\
D_1 & D_2 & D_3 & C
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
p
\end{bmatrix}
=
\begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
g
\end{bmatrix}
\]

for brevity:

\[
\begin{bmatrix}
Q & G \\
D & C
\end{bmatrix}
\begin{bmatrix}
f \\
g
\end{bmatrix}
\]

with \(Q_1 = Q_2 = Q_3\).

⇒ Solve system with FGMRES and SIMPLE-type preconditioner
Turbulence equations \((k-\omega\) model) remain segregated
**SIMPLE-method**

Given $u^k$ and $p^k$:

1. solve $Qu^* = f - Gp^k$
2. solve $(C - DQ^{-1}G)p' = g - Du^* - Cp^k$
3. compute $u' = -Q^{-1}Gp'$
4. update $u^{k+1} = u^* + u'$ and $p^{k+1} = p^k + p'$

with the SIMPLE approximation $Q^{-1} \approx \text{diag}(Q)^{-1}$.

⇒ “Matrix-free”: only assembly and storage of $Q$ and $(C - DQ^{-1}G)$. For $D$, $G$ and $C$ the action suffices.
SIMPLER: additional pressure prediction

Given $u^k$ and $p^k$, start with a pressure prediction:

1. solve
   \[(C - D \text{diag}(Q)^{-1}G)p^* = g - Du^k - D \text{diag}(Q)^{-1}(f - Qu^k)\]

2. continue with SIMPLE using $p^*$ instead of $p^k$
Some practical constraints

Compact stencils are preferred on unstructured grids:
- neighbors of cell readily available; neighbors of neighbors not

Also preferred because of MPI parallel computation:
- domain decomposition, communication

Compact stencil?
- ✓ Matrix $Q_1 = Q_2 = Q_3$, thanks to defect correction
- ✗ Stabilization matrix $C$

⇒ modify SIMPLE(R) such that $C$ is not required on the l.h.s.
Treatment of stabilization matrix

- In SIMPLE, neglect $C$ in l.h.s. of pressure correction equation

\[
(C - D \text{diag}(Q)^{-1} G)p' = g - Du^* - Cp^k
\]

\[
\downarrow
\]

\[
-D \text{diag}(Q)^{-1} Gp' = g - Du^* - Cp^k
\]

- In SIMPLER, do not involve the mass equation when deriving the pressure prediction $p^*$

\[
(C - D \text{diag}(Q)^{-1} G)p^* = g - Du^k - D \text{diag}(Q)^{-1}(f - Qu^k)
\]

\[
\downarrow
\]

\[
-D \text{diag}(Q)^{-1} Gp^* = -D \text{diag}(Q)^{-1}(f - Qu^k)
\]
## Container vessel

Tables show number of non-linear iterations and wall clock time needed to converge to machine precision, starting from uniform flow.

Model-scale $\text{Re} = 1.3 \cdot 10^7$, max cell aspect ratio $1 : 1600$

<table>
<thead>
<tr>
<th>grid</th>
<th>CPU cores</th>
<th>SIMPLE</th>
<th>KRYLOV-SIMPLER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td># its</td>
<td>Wall clock</td>
</tr>
<tr>
<td>13.3m</td>
<td>128</td>
<td>3187</td>
<td>5h 26mn</td>
</tr>
</tbody>
</table>
## Tanker

Model-scale $Re = 4.6 \cdot 10^6$, max cell aspect ratio 1 : 7000

<table>
<thead>
<tr>
<th>grid</th>
<th>CPU cores</th>
<th>SIMPLE</th>
<th>KRYLOV-SIMPLER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>its</td>
<td>Wall clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>its</td>
<td>Wall clock</td>
</tr>
<tr>
<td>0.25m</td>
<td>8</td>
<td>1379</td>
<td>25mn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>316</td>
<td>29mn</td>
</tr>
<tr>
<td>0.5m</td>
<td>16</td>
<td>1690</td>
<td>37mn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>271</td>
<td>25mn</td>
</tr>
<tr>
<td>1m</td>
<td>32</td>
<td>2442</td>
<td>57mn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>303</td>
<td>35mn</td>
</tr>
<tr>
<td>2m</td>
<td>64</td>
<td>3534</td>
<td>1h 29mn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>519</td>
<td>51mn</td>
</tr>
</tbody>
</table>

Full-scale $Re = 2.0 \cdot 10^9$, max cell aspect ratio 1 : 930 000

<table>
<thead>
<tr>
<th>grid</th>
<th>CPU cores</th>
<th>SIMPLE</th>
<th>KRYLOV-SIMPLER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>its</td>
<td>Wall clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>its</td>
<td>Wall clock</td>
</tr>
<tr>
<td>2.7m</td>
<td>64</td>
<td>29578</td>
<td>16h 37mn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1330</td>
<td>3h 05mn</td>
</tr>
</tbody>
</table>
7. Conclusions

- **MSIMPLER** is at present the fastest of all SIMPLE-type preconditioners.
- In our experiments, MSIMPLER proved to be cheaper than SILU, especially when the problem is solved with high accuracy.
- MSIMPLER shows better performance than LSC. Both have similar convergence characteristics.
- For academic problems, Modified Augmented Lagrangian (MAL) and grad-div are nearly independent of the grid size and Reynolds number.
- MAL/grad-div are faster than (M)SIMPLER
- Future research: MAL/grad-div for industrial (Maritime) applications
References

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


★ X. He and C. Vuik Comparison of some preconditioners for incompressible Navier-Stokes equations Delft University of Technology Delft Institute of Applied Mathematics Report 13-10