New Block Preconditioners for Saddle-Point Problems

Michele Benzi

Department of Mathematics and Computer Science
Emory University
Atlanta, GA 30322, USA

Workshop on Recent Developments in the Solution of Indefinite Systems
TU Eindhoven, 17 April 2012
Acknowledgments

- Michael Ng and Qiang Niu (HKBU, Hong Kong)
- Zhen Wang (Emory, now ORNL)
- Gwenol Grandperrin (EPFL, Switzerland)

Outline

- Problem description
- A Relaxed Dimensional Factorization Preconditioner
- Eigenvalue analysis
- Parameter choice by Fourier analysis
- Numerical examples
- An application to Hemodynamics
- Conclusions
The Problem

We consider the incompressible Navier–Stokes equations:

\[
\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} - \nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p &= \mathbf{f} \quad \text{on } \Omega \times (0, T], \\
\nabla \cdot \mathbf{u} &= 0 \quad \text{on } \Omega \times [0, T], \\
\mathbf{u} &= \mathbf{g} \quad \text{on } \partial \Omega \times [0, T], \\
\mathbf{u}(x, 0) &= \mathbf{u}_0(x) \quad \text{on } \Omega
\end{aligned}
\]

on an open bounded domain \( \Omega \subset \mathbb{R}^d \) \((d = 2, 3)\), where \( \mathbf{u} = \mathbf{u}(x, t) \) and \( p = p(x, t) \) are the unknown velocity and pressure field, \( \nu \) is the kinematic viscosity and \( \mathbf{f}, \mathbf{g} \) and \( \mathbf{u}_0 \) are given functions.
Time discretization together with a spatial discretization by finite elements or finite differences leads – upon linearization – to a sequence of (large and sparse) linear systems of the form

$$Hx = b,$$

with

$$H = \begin{pmatrix} A & B^T \\ -B & 0 \end{pmatrix}, \quad x = \begin{pmatrix} u \\ p \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} f \\ -g \end{pmatrix}.$$

This is a (generalized) saddle point problem.
Solving the saddle point problem

- An efficient and robust preconditioner is needed if this system is solved using a Krylov subspace method.
- “Efficient and robust” means fast convergence that is relatively independent of mesh size, viscosity, etc.
- Existing preconditioners for this type of saddle-point problem include:
  - Block diagonal and block triangular preconditioners (Elman, Silvester & Wathen, 2005)
  - Augmented Lagrangian-based preconditioners (B., Olshanskii & Wang, 2006/2011)
  - Preconditioners based on Dimensional Splitting (DS) (B. & Guo, 2011)
  - Relaxed Dimensional Factorization preconditioner (RDF) (B., Ng, Niu & Wang, 2011)
When Picard linearization is used, the matrix $H$ has the form (in 2D)

$$
H = \begin{pmatrix}
A_1 & 0 & B_1^T \\
0 & A_2 & B_2^T \\
-B_1 & -B_2 & 0
\end{pmatrix},
$$

with $A_1 \in \mathbb{R}^{n_1 \times n_1}$, $A_2 \in \mathbb{R}^{n_2 \times n_2}$ and $B_i \in \mathbb{R}^{m \times n_i}$, $i = 1, 2$.

The dimensional splitting of $H$ is given by

$$
H = \begin{pmatrix}
A_1 & 0 & B_1^T \\
0 & 0 & 0 \\
-B_1 & 0 & 0
\end{pmatrix} + \begin{pmatrix}
0 & 0 & 0 \\
0 & A_2 & B_2^T \\
0 & -B_2 & 0
\end{pmatrix} = H_1 + H_2.
$$
The DS preconditioner is defined as $P = \frac{1}{\alpha} (H_1 + \alpha I)(H_2 + \alpha I)$, or

$$P = \frac{1}{\alpha} \begin{pmatrix} A_1 + \alpha I & 0 & B_1^T \\ 0 & \alpha I & 0 \\ -B_1 & 0 & \alpha I \end{pmatrix} \begin{pmatrix} \alpha I & 0 & 0 \\ 0 & A_2 + \alpha I & B_2^T \\ 0 & -B_2 & \alpha I \end{pmatrix} = \begin{pmatrix} \alpha I + A_1 & -\frac{1}{\alpha} B_1^T B_2 & B_1^T \\ 0 & \alpha I + A_2 & B_2^T \\ -B_1 & -B_2 & \alpha I \end{pmatrix}.$$ 

Here $\alpha > 0$ is a parameter.
The difference between the preconditioner and the coefficient matrix is

\[
P - H = \begin{pmatrix}
\alpha I & -\frac{1}{\alpha} B_1^T B_2 & 0 \\
0 & \alpha I & 0 \\
0 & 0 & \alpha I
\end{pmatrix}
\]

As \( \alpha \to 0 \), the diagonal entries vanish, but the off-diagonal block blows up.

Choice of \( \alpha \) requires a trade-off.
An improved variant of the DS preconditioner may be constructed as follows:

\[
M = \frac{1}{\alpha} \begin{pmatrix} A_1 & 0 & B_1^T \\ 0 & \alpha l & 0 \\ -B_1 & 0 & \alpha l \end{pmatrix} \begin{pmatrix} \alpha l & 0 & 0 \\ 0 & A_2 & B_2^T \\ 0 & -B_2 & \alpha l \end{pmatrix}
\]

\[
= \begin{pmatrix} A_1 & -\frac{1}{\alpha} B_1^T B_2 & B_1^T \\ 0 & A_2 & B_2^T \\ -B_1 & -B_2 & \alpha l \end{pmatrix}.
\]

This is the Relaxed Dimensional Factorization preconditioner (RDF).
The difference between this preconditioner and the coefficient matrix is

\[
R = M - H = \begin{pmatrix}
0 & -\frac{1}{\alpha} B_1^T B_2 & 0 \\
0 & 0 & 0 \\
0 & 0 & \alpha I \\
\end{pmatrix}
\]

This suggests a better performance of RDF, since more blocks are zero and the remaining nonzero blocks are the same as with DS.
**Theorem 1.** The preconditioned matrix $T = H M^{-1}$ has an eigenvalue at $\lambda = 1$ with multiplicity $n_1 + n_2$. The remaining $m$ eigenvalues are the eigenvalues $\mu_i$ of the matrix

$$Z_\alpha = \alpha^{-1}(S_1 + S_2) - 2\alpha^{-2}S_1S_2,$$

where

$$S_1 = B_1(A_1 + \alpha^{-1}B_1^TB_1)^{-1}B_1^T$$

and

$$S_2 = B_2(A_2 + \alpha^{-1}B_2^TB_2)^{-1}B_2^T.$$
Theorem 2. The eigenvalues $\mu_i$ of $Z_\alpha$ are of the form

$$\mu_i = \frac{\alpha \lambda_i}{1 + \alpha \lambda_i}, \quad i = 1 : m,$$

where the $\lambda_i$'s satisfy the generalized eigenproblem

$$BA^{-1}B^T \phi_i = \lambda_i(\alpha^2 I + \hat{S}_1 \hat{S}_2)\phi_i,$$

with $\hat{S}_k = B_k A_k^{-1} B_k^T$ ($k = 1, 2$).
Eigenvalues of the preconditioned Oseen matrix on a $32 \times 32$ grid with Q2-Q1 finite elements. Top: DS preconditioner, bottom: RDF preconditioner. Left: $\nu = 0.01$, Right: $\nu = 0.001$. 
Theorem 2 can be used to estimate the magnitude of the eigenvalues \( \lambda \neq 1 \).

It can be shown that they go to zero like \( O(\alpha) \) for \( \alpha \to 0^+ \)
and like \( O(\alpha^{-1}) \) for \( \alpha \to \infty \).

In practice, diagonal scaling is applied to \( H \) before forming the RDF preconditioner. This significantly improves performance.
Applying the RDF preconditioner

Let $\hat{A}_i := A_i + \alpha^{-1} B_i^T B_i$ ($i = 1, 2$). The RDF preconditioner can be factored as

$$
M = \begin{pmatrix}
A_1 & 0 & B_1^T/\alpha \\
0 & I & 0 \\
-B_1 & 0 & I
\end{pmatrix}
\begin{pmatrix}
I & 0 & 0 \\
0 & A_2 & B_2^T \\
0 & -B_2 & \alpha I
\end{pmatrix}

= \begin{pmatrix}
I & 0 & B_1^T/\alpha \\
0 & I & 0 \\
0 & 0 & I
\end{pmatrix}
\begin{pmatrix}
\hat{A}_1 & 0 & 0 \\
0 & I & 0 \\
-B_1 & 0 & I
\end{pmatrix}
\begin{pmatrix}
I & 0 & 0 \\
0 & \hat{A}_2 & B_2^T \\
0 & 0 & \alpha I
\end{pmatrix}
\begin{pmatrix}
I & 0 & 0 \\
0 & I & 0 \\
0 & -B_2/\alpha & I
\end{pmatrix},
$$

so only subsystems involving $\hat{A}_1$ and $\hat{A}_2$ need to be solved. Moreover, these solves can be done inexactly.

Note: 3D case is analogous.
Estimating the optimal $\alpha$

- This is done using Fourier analysis

- We make the usual simplifying assumptions:
  - the PDE problem has constant coefficients
  - the problem is defined on the unit square/cube with periodic boundary conditions
  - the grid is uniform
  - the matrices $A_1, A_2, B_1, B_2$ are all the same size and commute
Under these assumptions, $A_1, A_2, B_1, B_2$ are all diagonalized by the discrete Fourier transform; we denote their generic eigenvalues by $a_1, a_2, b_1, b_2$.

From this we obtain that $Z_\alpha$ is also diagonalized by the discrete Fourier transform and the eigenvalues of $Z_\alpha$ are given by

$$\lambda(\alpha) = \frac{(s_1 + s_2)}{\alpha} - \frac{2s_1 s_2}{\alpha^2}$$

with

$$s_1 = \frac{b_1^2}{a_1 + b_1^2/\alpha}$$
$$s_2 = \frac{b_2^2}{a_2 + b_2^2/\alpha}.$$
For a finite difference discretization, we have

\[ a_1 = a_2 = \nu(2-e^{i2\pi h\theta} - e^{i2\pi h\theta}) + h(e^{i2\pi h\theta} - e^{-i2\pi h\theta}), \quad \theta = 1 : K \]

and

\[ b_1 = b_2 = h(1 - e^{-i2\pi h\theta}), \quad \theta = 1 : K. \]

Following Theorem 1, we try to maximize clustering of the eigenvalues of \( Z_\alpha \) around 1.

To this end, we find the \( \alpha \) that minimizes the average distance of the eigenvalues of \( Z_\alpha \) from 1. Note that this is an off-line computation.

Numerical experiments show that performance is not overly sensitive w.r.t. \( \alpha \)
Numerical experiments on model problems

- Mostly on a steady 2D lid-driven cavity discretized by Q2-Q1 or Q2-P1 finite elements using the MATLAB package IFISS
- Viscosity values $\nu = 0.1$, $\nu = 0.01$, $\nu = 0.001$
- Experiments are performed on $16 \times 16$, $32 \times 32$, $64 \times 64$ and $128 \times 128$ (uniform and stretched) grids
- The preconditioner is applied as a right preconditioner with restarted GMRES and maximum subspace dimension 20
- The linear solver stops when the relative residual drops below $10^{-6}$
Leaky lid-driven cavity

Top: $\nu = 0.1$, middle: $\nu = 0.01$, bottom: $\nu = 0.001$:

<table>
<thead>
<tr>
<th>Grid</th>
<th>DS optimal</th>
<th>RDF optimal</th>
<th>RDF FA estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_{opt}$</td>
<td>$\alpha_{opt}$</td>
<td>$\alpha_F$</td>
</tr>
<tr>
<td></td>
<td>$its$</td>
<td>$its$</td>
<td>$its$</td>
</tr>
<tr>
<td>16 × 16</td>
<td>0.03</td>
<td>0.05</td>
<td>0.07</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>32 × 32</td>
<td>0.009</td>
<td>0.01</td>
<td>0.025</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>64 × 64</td>
<td>0.002</td>
<td>0.005</td>
<td>0.008</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>128 × 128</td>
<td>0.0005</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grid</th>
<th>DS optimal</th>
<th>RDF optimal</th>
<th>RDF FA estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_{opt}$</td>
<td>$\alpha_{opt}$</td>
<td>$\alpha_F$</td>
</tr>
<tr>
<td></td>
<td>$its$</td>
<td>$its$</td>
<td>$its$</td>
</tr>
<tr>
<td>16 × 16</td>
<td>0.17</td>
<td>0.30</td>
<td>0.12</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>32 × 32</td>
<td>0.06</td>
<td>0.10</td>
<td>0.057</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>64 × 64</td>
<td>0.015</td>
<td>0.025</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>128 × 128</td>
<td>0.005</td>
<td>0.007</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grid</th>
<th>DS optimal</th>
<th>RDF optimal</th>
<th>RDF FA estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_{opt}$</td>
<td>$\alpha_{opt}$</td>
<td>$\alpha_F$</td>
</tr>
<tr>
<td></td>
<td>$its$</td>
<td>$its$</td>
<td>$its$</td>
</tr>
<tr>
<td>16 × 16</td>
<td>0.40</td>
<td>0.50</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>22</td>
<td>52</td>
</tr>
<tr>
<td>32 × 32</td>
<td>0.18</td>
<td>0.20</td>
<td>0.063</td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>31</td>
<td>60</td>
</tr>
<tr>
<td>64 × 64</td>
<td>0.07</td>
<td>0.05</td>
<td>0.032</td>
</tr>
<tr>
<td></td>
<td>37</td>
<td>27</td>
<td>31</td>
</tr>
<tr>
<td>128 × 128</td>
<td>0.02</td>
<td>0.03</td>
<td>0.015</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>24</td>
<td>28</td>
</tr>
</tbody>
</table>
Using inexact solves

So far, the subsystems involving $\hat{A}_i = A_i + \alpha^{-1}B_i^T B_i$, $i = 1, 2$ were solved exactly using a sparse direct solver.

We may instead use an **inner iterative solver** to find an inexact solution and reduce overall costs.

In the next experiment, GMRES(20) preconditioned by ILU(0) is used as an inner solver (i.e., an inexact solver for $\hat{A}_1, \hat{A}_2$).

FGMRES with right RDF preconditioning is used as the outer solver.
Comparison of the convergence curves of the exact and inexact inner solver tested on the steady Oseen problem with \( \nu = 0.01 \), left: 64 \( \times \) 64 grid, right: 128 \( \times \) 128 grid
Inexact solve

The same comparison with $\nu = 0.001$, left: $64 \times 64$ grid, right: $128 \times 128$ grid
This table shows a comparison of the CPU time of exact vs. inexact inner solvers, RDF preconditioned GMRES(20) or FGMRES, tested on the Oseen problem with $\nu = 0.001$, Q2-Q1 finite elements.

<table>
<thead>
<tr>
<th>Grid</th>
<th>$tol_{in} = 0.1$</th>
<th>$tol_{in} = 0.05$</th>
<th>Exact</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$its$</td>
<td>$cpu$</td>
<td>$its$</td>
</tr>
<tr>
<td>16 x 16</td>
<td>24</td>
<td>0.27</td>
<td>23</td>
</tr>
<tr>
<td>32 x 32</td>
<td>33</td>
<td>0.62</td>
<td>32</td>
</tr>
<tr>
<td>64 x 64</td>
<td>31</td>
<td>1.91</td>
<td>30</td>
</tr>
<tr>
<td>128 x 128</td>
<td>28</td>
<td>12.38</td>
<td>27</td>
</tr>
</tbody>
</table>
In this example, the steady Oseen problem is discretized by a Marker-and-Cell (finite difference) scheme and solved with GMRES(50) with RDF preconditioner.

The subproblems with $\hat{A}_i$ are solved inexactly by an algebraic multigrid V-cycle, with symmetric Gauss-Seidel smoother (code: MI20 by Boyle, Mihajlovic & Scott).
A 3D experiment: results

<table>
<thead>
<tr>
<th>Grid</th>
<th>$\nu = 0.1$</th>
<th>$\nu = 0.01$</th>
<th>$\nu = 0.005$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$i ts$ ($\alpha_{opt}$)</td>
<td>$i ts$ ($\alpha_{opt}$)</td>
<td>$i ts$ ($\alpha_{opt}$)</td>
</tr>
<tr>
<td>$16 \times 16 \times 16$</td>
<td>12 (1.6-3.6)</td>
<td>18 (13.1-14)</td>
<td>27 (21.6-21.9)</td>
</tr>
<tr>
<td>Setup time</td>
<td>0.11</td>
<td>0.48</td>
<td>0.13</td>
</tr>
<tr>
<td>Iter time</td>
<td>0.58</td>
<td>1.32</td>
<td>2.30</td>
</tr>
<tr>
<td>Total time</td>
<td>0.69</td>
<td>1.80</td>
<td>2.43</td>
</tr>
<tr>
<td>$32 \times 32 \times 32$</td>
<td>12 (2.3-2.4)</td>
<td>19 (12.3-14)</td>
<td>26 (20.5 - 22)</td>
</tr>
<tr>
<td>Setup time</td>
<td>1.60</td>
<td>1.89</td>
<td>2.45</td>
</tr>
<tr>
<td>Iter time</td>
<td>10.87</td>
<td>17.73</td>
<td>36.27</td>
</tr>
<tr>
<td>Total time</td>
<td>12.47</td>
<td>19.62</td>
<td>38.72</td>
</tr>
<tr>
<td>$48 \times 48 \times 48$</td>
<td>13 (1.7-3.0)</td>
<td>19 (12.6-14)</td>
<td>26 (20.4 - 22)</td>
</tr>
<tr>
<td>Setup time</td>
<td>7.04</td>
<td>9.43</td>
<td>8.61</td>
</tr>
<tr>
<td>Iter time</td>
<td>45.92</td>
<td>88.39</td>
<td>125.19</td>
</tr>
<tr>
<td>Total time</td>
<td>52.96</td>
<td>97.82</td>
<td>133.80</td>
</tr>
<tr>
<td>$64 \times 64 \times 64$</td>
<td>13 (2.1-2.6)</td>
<td>19 (12.7-14)</td>
<td>26 (20.6 - 22)</td>
</tr>
<tr>
<td>Setup time</td>
<td>18.24</td>
<td>23.46</td>
<td>25.22</td>
</tr>
<tr>
<td>Iter time</td>
<td>112.06</td>
<td>219.17</td>
<td>333.51</td>
</tr>
<tr>
<td>Total time</td>
<td>130.30</td>
<td>242.63</td>
<td>358.73</td>
</tr>
</tbody>
</table>
Numerical results with LifeV

LifeV
http://www.lifev.org

LifeV relies on several external libraries:

- ParMetis/Metis for parallel mesh partitioning;
- Trilinos (10.8) for matrix and vector parallel distribution, for parallel solvers, and for parallel preconditioners;
- Boost, SuiteSparse (UMFPACK), HDF5.
The simulations were run on the CADMOS IBM Blue Gene/P at EPFL, Lausanne, Switzerland.

- 4 racks, one row, wired as a 16x16x16 3D torus
- 4096 quad-core nodes, PowerPC 450, 850 MHz
- Energy efficient, water cooled
- 56 Tflops peak, 46 Tflops LINPACK
- 16 TB of memory (4 GB per compute node)
- 1 PB of disk space, GPFS parallel file system
- OS Linux SuSE SLES 10
Hemodynamic problem (simulation of cerebral aneurysm)

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Velocity DoFs</th>
<th>Pressure DoFs</th>
<th>$h_{min}$</th>
<th>$h_{max}$</th>
<th>$h_{av}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coarse</td>
<td>597,093</td>
<td>27,242</td>
<td>0.015</td>
<td>0.059</td>
<td>0.035</td>
</tr>
<tr>
<td>Fine</td>
<td>4,557,963</td>
<td>199,031</td>
<td>0.005</td>
<td>0.051</td>
<td>0.018</td>
</tr>
</tbody>
</table>
Boundary conditions:

\[ u = 0 \text{ on } \Gamma_{wall}, \quad u = \varphi_{flux}n \text{ on } \Gamma_{in}, \quad \nu \frac{\partial u}{\partial n} - pn = 0 \text{ on } \Gamma_{out}. \]
Hemodynamic problem (cont.)

We impose the following inflow:

\[ \varphi_{\text{flux}}(t) = a_0 + \sum_{i=1}^{7} a_k \cos \left( \frac{2\pi k t}{T} \right) + b_k \sin \left( \frac{2\pi k t}{T} \right) \]

Other parameter settings:

▶ Unsteady problem, $\Delta t = 10^{-5}$
▶ Nonlinear relative residual tolerance: $10^{-6}$
▶ Outer (F)GMRES relative residual tolerance: $10^{-6}$
▶ Inner GMRES relative residual tolerance: 0.05
▶ Inner GMRES sub-iterations preconditioned with ML (Trilinos)
▶ Optimal $\alpha$ chosen experimentally

Caveat: The following results are very preliminary!
Hemodynamic problem (cont.)

<table>
<thead>
<tr>
<th>( \nu )</th>
<th>Cores</th>
<th>Coarse mesh</th>
<th></th>
<th>Fine mesh</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iter</td>
<td>Setup time</td>
<td>Iter time</td>
<td>Iter</td>
</tr>
<tr>
<td>0.28</td>
<td>128</td>
<td>5.0</td>
<td>8.38</td>
<td>4.58</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>6.0</td>
<td>5.92</td>
<td>3.88</td>
<td>8.0</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>6.0</td>
<td>4.99</td>
<td>3.08</td>
<td>8.0</td>
</tr>
<tr>
<td>0.035</td>
<td>128</td>
<td>6.0</td>
<td>8.45</td>
<td>7.83</td>
<td>11.0</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>6.5</td>
<td>5.96</td>
<td>10.11</td>
<td>14.0</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>7.0</td>
<td>4.99</td>
<td>31.05</td>
<td>17.0</td>
</tr>
</tbody>
</table>

**Table:** Results for RDF preconditioner (averages per Picard iteration)

**Note:** It is unclear why the number of iteration changes as the number of cores increases; since the preconditioner is the same, it should remain constant. We are working to understand what the problem is.
### Table: Result for PCD preconditioner (averages per Picard iteration)

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>Cores</th>
<th>Coarse mesh</th>
<th>Fine mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iter</td>
<td>Setup time</td>
</tr>
<tr>
<td>0.28</td>
<td>128</td>
<td>28.0</td>
<td>10.28</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>33.0</td>
<td>4.08</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>34.0</td>
<td>2.00</td>
</tr>
<tr>
<td>0.035</td>
<td>128</td>
<td>29.0</td>
<td>10.28</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>34.0</td>
<td>4.07</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>35.0</td>
<td>2.00</td>
</tr>
</tbody>
</table>

**Note:** Here a different multi-level solver is used to apply the preconditioner (a two-level, non-overlapping additive Schwarz preconditioner for the convection-diffusion solves, and ML-PCG for the Poisson solves).
Conclusions and an open question

- Numerical experiments indicate fast, mesh-independent convergence of GMRES preconditioned with RDF.
- Convergence rate is only moderately affected by decreasing viscosity $\nu$; RDF is competitive with other state-of-the-art preconditioners like PCD.
- Stretched grids not a problem (see JCP paper).
- Fast convergence rates of RDF are maintained even when the inner solves are performed with low accuracy.
- Very good results also for stabilized FEM (see Zhen Wang’s PhD thesis).
- How to select $\alpha$ for realistic problems?