Solving the linearized Navier–Stokes equations using semi-Toeplitz preconditioning

Samuel Sundberg

Information Technology, Department of Scientific Computing,
Uppsala University,
Box 337, SE-751 05 Uppsala, Sweden,
Samuel.Sundberg@tdb.uu.se
WWW home page: http://www.tdb.uu.se/

Abstract. A semi-Toeplitz preconditioner for the linearized Navier–Stokes equation for compressible flow is proposed and tested. The preconditioner is applied to the linear system of equations to be solved in each time step of an implicit method. The equations are solved with flat plate boundary conditions and are linearized around the Blasius solution. The grids are stretched in the normal direction to the plate and the quotient between the time step and the space step is varied. The preconditioner works well in all tested cases and outperforms the method without preconditioning both in number of iterations and execution time.

1 Introduction

Iterative methods for solving large sparse systems of equations are frequently used in time dependent compressible flow problems. The equations are usually integrated in time by explicit time stepping methods but in certain flow situations implicit methods are chosen. This is the case when there are several timescales present in the flow, as in low Mach number flows, and when the cell size is small in order to resolve all the spatial scales. In these cases explicit time stepping is very inefficient due to restrictive stability requirements, and instead we use implicit methods. The discretized equations are linearized and solved by a Newton-like iteration. This approach gives us a system,

$$Au = b, \tag{1.1}$$

that has to be solved at least once in every time step. The equation (1.1) is a large, sparse system that is unsuitable for direct methods in more spatial dimensions than one due to problems with fill-in, and therefore iterative methods are preferred.

Sometimes iterative methods fail to deliver the solution at a resonable cost: the convergence is slow or is not obtained at all. In these cases we can use preconditioning of the system (1.1) to improve the convergence rate of the iterative method. In this paper, we continue working with the semi-Toeplitz preconditioner that was defined and analyzed for a scalar equation in [12]. The results of [12] clearly show that the semi-Toeplitz preconditioner works very well for a scalar model problem that mimics the behaviour of boundary layers in fluid dynamics. In order to verify the applicability of these results for a larger class of problems we now investigate the linearized Navier–Stokes equations for flow over a flat plate.

Semi-Toeplitz and semicirculant preconditioners have been developed for finite difference discretizations of hyperbolic partial differential equations in [4], [5], [7], [8]. A semicirculant preconditioner is applied to the steady state solution of the linearized Navier–Stokes equations in [6]. The system of equations is first preconditioned by a semicirculant matrix. Then the forward Euler method is used in [6] to advance the solution in pseudo-time. The method is compared to standard explicit Runge-Kutta time stepping to reach the steady state and multigrid iteration in numerical experiments. The suggested method is superior especially for high Reynolds numbers Re.

The rest of the paper is organized as follows. In Section 2 we present the boundary layer equations and how the linearized Navier-Stokes equations are derived. Then in Section 3 we describe in detail the discretization we have used, and continue in Section 4 by introducing the preconditioner. Computational results appear in section 5 and finally we draw some conclusions in section 6. The norm in the paper is the Euclidean vector norm.

2 The linearized Navier-Stokes equations

Let $w = (\rho, \rho u, \rho v, \rho E)^T$ be the state vector, where ρ is the density, u and v are the velocity components in the x_1 and x_2 directions, and E is the total energy. Then the Navier-Stokes equations in two dimensions for the time-independent flow of a compressible fluid are in conservation form

$$\frac{\partial w}{\partial t} + \frac{\partial f(w, \nabla w)}{\partial x_1} + \frac{\partial g(w, \nabla w)}{\partial x_2} = 0. \tag{2.1}$$

The flux vectors f and g are defined by

$$f = \begin{pmatrix} \rho u \\ \rho u^2 + p - \mu S_{11} \\ \rho uv - \mu S_{21} \\ (\rho E + p)u - \mu(uS_{11} + vS_{21}) \end{pmatrix}, g = \begin{pmatrix} \rho v \\ \rho uv - \mu S_{12} \\ \rho v^2 + p - \mu S_{22} \\ (\rho E + p)v - \mu(uS_{12} + vS_{22}) \end{pmatrix}, \tag{2.2}$$

where μ is the viscosity coefficient and S is the strain rate tensor defined by

$$S = \begin{pmatrix} \frac{4}{3}u_{x_1} - \frac{2}{3}v_{x_2} & u_{x_2} + v_{x_1} \\ u_{x_2} + v_{x_1} & \frac{4}{3}v_{x_2} - \frac{2}{3}u_{x_1} \end{pmatrix}. \tag{2.3}$$

A subscript x_1 or x_2 denotes differentiation with respect to the variable. The pressure p satisfies

$$p = (\gamma - 1)\rho(E - 0.5(u^2 + v^2))$$

and the gas constant $\gamma = 1.4$. The Reynolds number is defined by $Re = \rho_\infty u_\infty l/\mu$ for some length scale l and ρ_∞ and u_∞ are the density and free stream velocity at ∞.

We consider a standard problem in fluid dynamics, namely the flow of gas in the boundary layer over a flat plate. We obtain a good approximation of the flow by solving the boundary layer equations [10], [11],

$$\begin{aligned} u\frac{\partial u}{\partial x_1} + v\frac{\partial u}{\partial x_2} =& \nu\frac{\partial^2 u}{\partial x_2^2}, \\ \frac{\partial u}{\partial x_1} + \frac{\partial v}{\partial x_2} =& 0, \end{aligned} \tag{2.4}$$

where the kinematic viscosity is defined by $\nu = \mu/\rho$. The velocity vanishes at the surface, i.e. we have $u = v = 0$ at $x_2 = 0$ and $u = u_\infty =$.

Exploiting the transformation $\xi = 0.5x_2(u_\infty/\nu x_1)^{1/2}$, (2.4) is transformed into the Blasius equation

$$f''' + \frac{1}{2}ff'' = 0.$$

Here, the boundary conditions are $f = f' = 0$ and $f'' \approx 1.33$ at $\xi = 0$. This ordinary differential equation can be solved numerically, yielding the Blasius solution

$$\begin{aligned} \hat{u} &= \frac{1}{2}u_\infty f'(\xi), \\ \hat{v} &= \frac{1}{2}\left(\frac{\nu u_\infty}{x_1}\right)^{1/2}(\xi f'(\xi) - f(\xi)). \end{aligned} \tag{2.5}$$

By linearizing the Navier-Stokes equations around the solution (2.5), symmetrizing, and eliminating the pressure as in [6], we arrive at the following linear system of partial differential equations

$$\frac{\partial U}{\partial t} + A_1\frac{\partial U}{\partial x_1} + A_2\frac{\partial U}{\partial x_2} - B_1\frac{\partial^2 U}{\partial x_1^2} - B_2\frac{\partial^2 U}{\partial x_2^2} - B_3\frac{\partial^2 U}{\partial x_1 \partial x_2} = F(x_1, x_2, t), \tag{2.6}$$

where $U = (u\ v\ \rho)^T$ and F depends on the Blasius solution. The computational domain is chosen to be $1 \le x_1 \le 2$, and $0 \le x_2 \le 1$ with the plate at $x_2 = 0$.

The coefficients A_1, A_2, B_1, B_2, and B_3 are (3×3)-matrices of the form

$$A_1 = \begin{pmatrix} \hat{u} & 0 & 1 \\ 0 & \hat{u} & 0 \\ 1 & 0 & \hat{u} \end{pmatrix}, A_2 = \begin{pmatrix} \hat{v} & 0 & 0 \\ 0 & \hat{v} & 1 \\ 0 & 1 & \hat{v} \end{pmatrix},$$

$$B_1 = \nu\begin{pmatrix} \frac{4}{3} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, B_2 = \nu\begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{4}{3} & 0 \\ 0 & 0 & 0 \end{pmatrix}, B_3 = \nu\begin{pmatrix} 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

where $\hat{u}$ and $\hat{v}$ are given by (2.5).

3 Discretization

We let $u^n_{j,k}$ denote the approximate solution of U at the point $(x_{1,j}, x_{2,k})$ at the time $n\Delta t$. We use a grid with fixed step size in the x_1 direction and a stretching in the x_2 direction with a parameter $\eta \geq 0$,

$$\begin{aligned}
&x_{1,j} = 1 + (j - \tfrac{1}{2})\Delta x_1, \qquad j = 0, 1, \ldots, m_1 + 1, \qquad \Delta x_1 = \tfrac{1}{m_1},\\
&x_{2,k} = \left(\tfrac{e^{\eta(k-1)/m_2} - 1}{e^{\eta} - 1} + \tfrac{e^{\eta k/m_2} - 1}{e^{\eta} - 1} \right) \Big/ 2, \qquad k = 1, \ldots m_2\\
&x_{2,0} = -x_{2,1}, \quad x_{2,m_2+1} = 2 - x_{2,m_2}.
\end{aligned} \tag{3.1}$$

The number of inner grid points in the x_1-direction is m_1 and in the x_2-direction m_2. The points at $j = 0, k = 0, j = m_1 + 1$, and $k = m_2 + 1$ are 'ghost points' and used to introduce the boundary conditions, see below. The number of unknown vectors $u^n_{j,k}$ is $m_1 m_2$ in each time step.

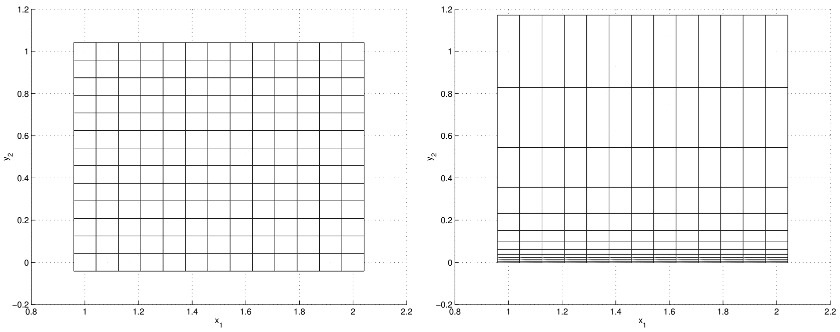

Fig. 1. The computational grid for $\eta = 0$ (left) and $\eta = 5$ (right). $m_1 = m_2 = 31$ for both figures.

We discretize equation (2.6) in space by finite difference approximations and in time using the second order trapezoidal rule with time step Δt

$$\begin{aligned}
&\frac{u^{n+1}_{j,k} - u^n_{j,k}}{\Delta t} + \frac{A_1}{2}\left(D_{0_j} u^{n+1}_{j,k} + D_{0_j} u^n_{j,k}\right) + \frac{A_2}{2}\left(D_{0_k} u^{n+1}_{j,k} + D_{0_k} u^n_{j,k}\right)\\
&- \frac{B_1}{2}\left((D_+D_-)_j\, u^{n+1}_{j,k} + (D_+D_-)_j\, u^n_{j,k}\right) - \frac{B_2}{2}\left((D_+D_-)_k\, u^{n+1}_{j,k} + (D_+D_-)_k\, u^n_{j,k}\right)\\
&- \frac{B_3}{2}\left(D_{0_k} D_{0_j} u^{n+1}_{j,k} + D_{0_k} D_{0_j} u^n_{j,k}\right) = \frac{1}{2}(F^{n+1} + F^n).
\end{aligned} \tag{3.2}$$

The spatial operators $(D_+D_-)_i$ and $(D_0)_i$ correspond to the usual difference approximations of derivatives of second order in the i-direction. They are given by

$$
\begin{aligned}
&D_+D_-u_i = a_i u_{i+1} + b_i u_i + c_i u_{i-1}, \\
&a_i = \frac{2}{h_{i^+}(h_{i^-} + h_{i^+})}, \quad b_i = -\frac{2}{h_{i^-}h_{i^+}}, \quad a_i = \frac{2}{h_{i^-}(h_{i^-} + h_{i^+})} \\
&D_0 u_i = d_i u_{i+1} + e_i u_i + f_i u_{i-1}, \\
&d_i = \frac{h_{i^-}}{h_{i^+}(h_{i^-} + h_{i^+})}, \quad e_i = \frac{h_{i^+} - h_{i^-}}{h_{i^-}h_{i^+}}, \quad f_i = -\frac{h_{i^+}}{h_{i^-}(h_{i^-} + h_{i^+})},
\end{aligned}
\tag{3.3}
$$

where h_{i^-} and h_{i^+} is the spatial step size in direction i before and after the value u_i. This gives us a compact nine point stencil for the spatial discretization. The resulting coefficent matrix is block tridiagonal.

In order to avoid spurious oscillations in the domain outside the boundary layer, artifical viscosity is added everywhere. The operators added to each of the three equations in both coordinate directions are given by

$$
D_{art,i} = \varepsilon \sigma h_i^3 (D_+D_-)_i^2 \tag{3.4}
$$

where σ is the absolute value of the velocity in the i direction as in [9] and ε is a small parameter. The terms are of $O(h^3)$ and do not destroy the formal accuracy of the discretization.

Given m_1, m_2, and η, the kinematic viscosity ν is chosen such that approximately half the grid points are inside the boundary layer. Then the solution in the layer is accurately resolved. In the direction parallel to the plate, the variation of the solution is much smaller than in the normal direction. It follows from [10] that the boundary layer ends at approximately $\xi = 3$. Thus, $0.5x_2\sqrt{u_\infty/\nu} = 3$ at $x_1 = 1$. In our case $u_\infty = 0.1$. Hence, the outer part of the boundary layer is at $x_{2bnd} \approx 20\sqrt{\nu}$. The corresponding grid point will now be located at

$$
\begin{aligned}
x_{2,m_2/2} &= 0.5\left(\frac{e^{\eta(m_2/2-1)/m_2} - 1}{e^\eta - 1} + \frac{e^{\eta/2} - 1}{e^\eta - 1}\right) \\
&\approx \frac{1}{e^{\eta/2} + 1} = 20\sqrt{\nu}.
\end{aligned}
\tag{3.5}
$$

Since $x_{2,m_2/2}$ is given by η and m_2, ν is now calculated from (3.5).

3.1 The boundary conditions

Numerical boundary conditions are determined by the Blasius solution at the inflow boundary to the left, the outflow boundary to the right, and the upper, open boundary. On the lower no-slip boundary, the flow is assumed to be zero and for all j we have for $k = 0$

$$
\begin{pmatrix} u_0 \\ v_0 \\ \rho_0 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ \rho_1 \end{pmatrix}.
$$

At the right boundary, we extrapolate the values in the two nearest points for all k

$$\begin{pmatrix} u_{m_1+1} \\ v_{m_1+1} \\ \rho_{m_1+1} \end{pmatrix} = \begin{pmatrix} 1\,0\,1 \\ 0\,2\,0 \\ 1\,0\,1 \end{pmatrix} \begin{pmatrix} u_{m_1} \\ v_{m_1} \\ \rho_{m_1} \end{pmatrix} - \frac{1}{2} \begin{pmatrix} 1\,0\,1 \\ 0\,2\,0 \\ 1\,0\,1 \end{pmatrix} \begin{pmatrix} u_{m_1-1} \\ v_{m_1-1} \\ \rho_{m_1-1} \end{pmatrix} + $$

$$+ \frac{1}{2} \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{u}(x_{1,m_1+1}) \\ \hat{v}(x_{1,m_1+1}) \\ \hat{\rho}(x_{1,m_1+1}) \end{pmatrix}, \quad (3.6)$$

where a hat-variable denotes the Blasius solution. At the left boundary where $j = 0$ we have for all k

$$\begin{pmatrix} u_0 \\ v_0 \\ \rho_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \\ \rho_1 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} u_2 \\ v_2 \\ \rho_2 \end{pmatrix} + $$

$$+ \frac{1}{2} \begin{pmatrix} 1\,0\,1 \\ 0\,2\,0 \\ 1\,0\,1 \end{pmatrix} \begin{pmatrix} \hat{u}(x_{1,0}) \\ \hat{v}(x_{1,0}) \\ \hat{\rho}(x_{1,0}) \end{pmatrix}. \quad (3.7)$$

At the upper boundary we have $k = m_2 + 1$ and for all j

$$\begin{pmatrix} u_{m_2+1} \\ v_{m_2+1} \\ \rho_{m_2+1} \end{pmatrix} = \begin{pmatrix} 2\,0\,0 \\ 0\,1\,1 \\ 0\,1\,1 \end{pmatrix} \begin{pmatrix} u_{m_2} \\ v_{m_2} \\ \rho_{m_2} \end{pmatrix} - \frac{1}{2} \begin{pmatrix} 2\,0\,0 \\ 0\,1\,1 \\ 0\,1\,0 \end{pmatrix} \begin{pmatrix} u_{m_2-1} \\ v_{m_2-1} \\ \rho_{m_2-1} \end{pmatrix} + $$

$$+ \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \hat{u}(x_{2,m_2+1}) \\ \hat{v}(x_{2,m_2+1}) \\ \hat{\rho}(x_{2,m_2+1}) \end{pmatrix}. \quad (3.8)$$

4 Preconditioning

We introduce left preconditioning of (1.1) with a preconditioner M,

$$M^{-1}Au = M^{-1}b. \quad (4.1)$$

In [1], Freund et al show that the residual reduction depends on the location of the eigenvalues of $M^{-1}A$ and the conditioning of its eigenvector matrix. For a scalar model problem we find analytic expressions for the eigenvalues of the preconditioned matrix on a uniform grid in [12]. Here we use similar semi-Toeplitz approximations to precondition the system of linear equations at time t^{n+1} resulting from the discretization of the linearized Navier–Stokes equations in (3.2).

We consider matrices T with a *Toeplitz* structure

$$T = I_{m_1} \otimes \alpha + R_{m_1} \otimes \beta, \tag{4.2}$$

where $\alpha, \beta \in \mathbb{R}^{3\times 3}$, I_{m_1} is the identity matrix of size $m_1 \times m_1$ and R_{m_1} of order m_1 is defined by

$$R_{m_1} = \begin{pmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & -1 & 0 \end{pmatrix}. \tag{4.3}$$

In the matrix R_{m_1} above, all elements outside the diagonals are zero. A *semi-Toeplitz* matrix is constructed by

$$M = \begin{pmatrix} T_{1,1} & T_{1,2} & & \\ T_{2,1} & \ddots & \ddots & \\ & \ddots & \ddots & T_{m_2-1,m_2} \\ & & T_{m_2,m_2-1} & T_{m_2,m_2} \end{pmatrix}, \tag{4.4}$$

where $T_{i,j}$ are all of the form defined in (4.2). The parameters α and β are computed using the process below.

The preconditioner is formed by discretizing the original equations with Dirichlet boundary conditions on the left and right boundaries to get the base matrix B. We then take the average over the sub- and superdiagonal entries in each block of B and over the diagonal entries in each block. Note that here all entries are 3×3 matrices and they are not constant due to the variable Blasius solution. This gives us

$$T_{i,j} = I_{m_1} \otimes \frac{1}{m_1} \sum_{k=1}^{m_1} [B_{i,j}]_{k,k} + R_{m_1} \otimes \frac{1}{2(m_1-1)} \sum_{k=1}^{m_1-1} \left([B_{i,j}]_{k,k+1} - [B_{i,j}]_{k+1,k} \right). \tag{4.5}$$

The preconditioner system, $Mx = y$, is solved using a Fast Modified Sine Transform (FMST) and the solution of narrowbanded systems and the total work is of order $\mathcal{O}(m_1 m_2 \log(m_1 m_2))$. This is done in three steps:

1. Perform Fast Fourier transforms of vectors $y \in \mathbb{C}^{\frac{m_1+1}{2}}$.
2. Solve $\frac{m_1+1}{2}$ tridiagonal systems of order $3m_2$.
3. Perform inverse Fast Fourier transforms of vectors $y \in \mathbb{C}^{\frac{m_1+1}{2}}$.

For further details concerning FMST and the preconditioner solve we refer to [3].

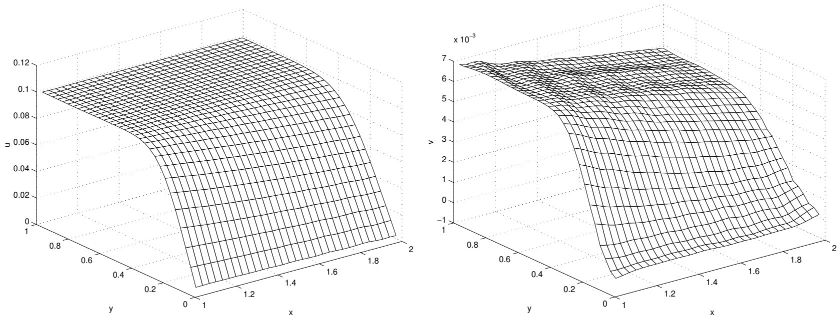

Fig. 2. Plot of steady state solution with u to the left and v to the right with $m_1 = m_2 = 31, \kappa = 10$ and $\eta = 0$.

5 Numerical results

The numerical experiments were executed on a Sun Fire 15k server with UltraSPARC III+ CPUs at 900 MHz, 8 MB of L2 Cache memory and 48 GB of primary RAM. During all experiments we have chosen the viscosity ν such that the boundary layer is resolved by half the gridpoints as in (3.5). The system is solved using restarted GMRES with a restarting length of 6, see [2]. The stopping criterion in the GMRES solver at iteration i was

$$\|M^{-1}(Au^{(i)} - b)\|/\|M^{-1}b\| \leq 10^{-6}.$$

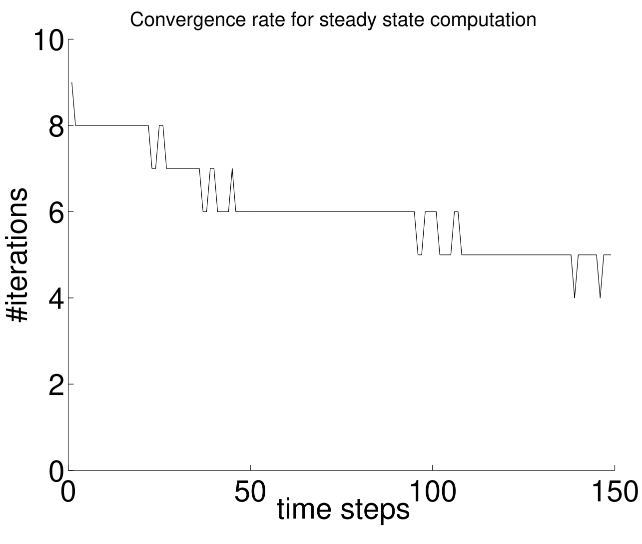

Fig. 3. Convergence history for the steady state computation.

In Fig. 2 we have two plots showing the result of a steady state computation. We used

$$\frac{\|u^n - u^{n-1}\|}{m_1 m_2 \Delta t} < \varepsilon_s$$

as a stopping criterion at time $n\Delta t$ and the definition of κ is $\Delta t/\Delta x_2$. For this computation we used $\varepsilon_s = 5 \cdot 10^{-4}$ requiring 149 time steps to reach the steady state. A plot of the number of iterations needed in each time step for GMRES(6) to solve the linear system is shown in Fig. 3, where we see that the number of iterations per step decreases as we approach steady state.

The rest of the results in this report are grid refinement studies where parameters that influence the convergence behaviour are investigated. For these studies we only solve the system of equations for the first time step and consider the convergence as we increase m_2 while keeping m_1 constant. Then ν will vary according to (3.5).

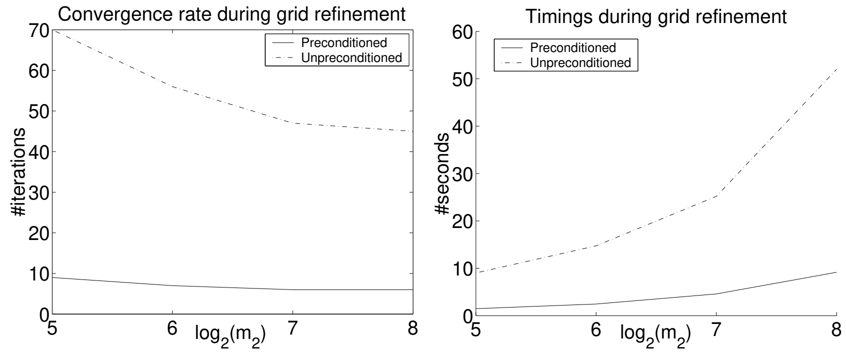

Fig. 4. A comparison between preconditioned and unpreconditioned GMRES(6) regarding iteration count (left) and timings (right) on a uniform grid with $\eta = 0, m_1 = 31$, and $\kappa = 10$.

In Fig. 4 we see a comparison between unpreconditioned GMRES(6) and preconditioned GMRES(6) with respect to iteration count (left) and runtime (right). As expected the preconditioned GMRES(6) shows a considerable reduction in the number of iterations needed for convergence. The standard GMRES(6) requires about seven times more iterations to reach convergence. Furthermore we see that the runtime for the preconditioned method is much smaller, even though it includes the extra work needed to form the preconditioner.

In Table 1 we present some numbers describing the performance in more detail for a grid without stretching ($\eta = 0$). The first column shows the quotient between the wall clock time for the program with and without preconditioning. We see that the preconditioned solver is roughly five times faster than the unpreconditioned solver. In the second column we present the time spent form-

$m_1 \times m_2$	t(prec)/t(noprec)	t(extra)/t(tot)	(t(p)/#it)/(t(up)/#it)
31x31	17%	7.3%	124%
31x63	18%	8.5%	131%
31x127	19%	9.3%	135%
31x255	19%	9.1%	131%

Table 1. Analysis of timings for one time step forming and computing with and without the preconditioner.

ing[1] the preconditioner as a percentage of the total time for the computation of one time step with the preconditioned solver. It appears that the percentage increases somewhat for the larger problems as there are less iterations needed for convergence. Since the matrices are independent of time, the preconditioner is created once before the first time step. Then it is reused in every time step. The third column compares the time for one iteration for the preconditioned method to the time for one iteration without preconditioning. The extra cost for solving the preconditioned system in each iteration is at most about one third of the total cost for the average GMRES(6) iteration.

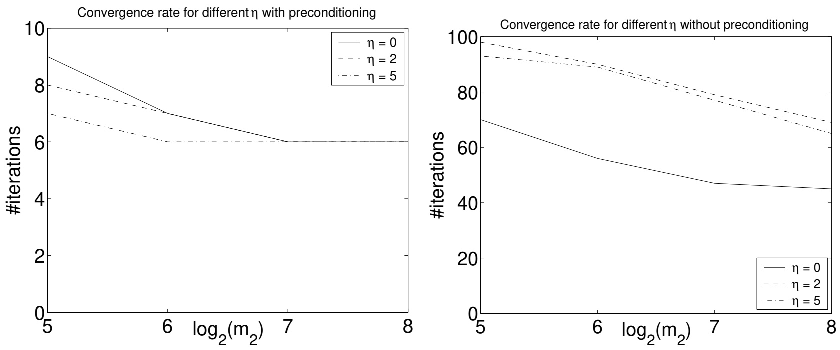

Fig. 5. Plots of the number of iterations for one time step with stretching in the x_2 direction with $\eta = 0, 2, 5$.

Another interesting aspect of the performance of the preconditioner is the convergence behaviour when a stretching in the grid is introduced. The grid is stretched only in the x_2-direction. The stretching is controlled by the factor η in equation (3.1) and the corresponding ν in (2.6) is calculated as in (3.5). As shown in Fig. 5, the preconditioned method yields about the same iteration count for the stretched grids as for the uniform grid and even a decrease for

[1] Initiating the matrix B in (4.5), forming the semi-Toeplitz approximation and setting up the trigonometric tables used for the preconditioned solver.

larger stretching factors. The unpreconditioned method requires a substantial increase in number of iterations to converge. Clearly the efficiency is improved by using the preconditioner in also this case.

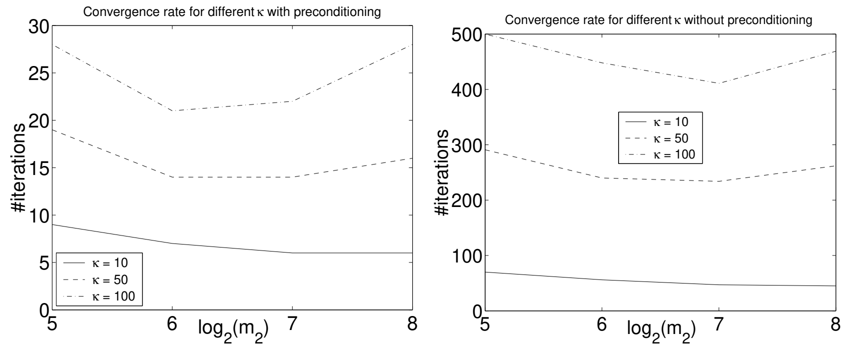

Fig. 6. Plots of number of iterations for different $\kappa = 10, 50, 100$

One of the reasons for using implicit methods in fluid computations is the unconditional stability of the trapezoidal method (or Crank-Nicolson method). Then the time steps can be much longer in cases where time resolution is not an issue. Therefore in Fig. 6 we study what happens when we use different values for $\kappa = \Delta t/\Delta x_2$ and $m_2 = 2^n - 1, \quad n = 5 \ldots 8, \eta = 0$. The trend is that we gain substantially by applying the preconditioner, especially for larger values of κ corresponding to stiff problems. The quotient between the number of iterations for the unpreconditioned solver versus the preconditioned one grows from 7.5 to 16.75 for the finest grid.

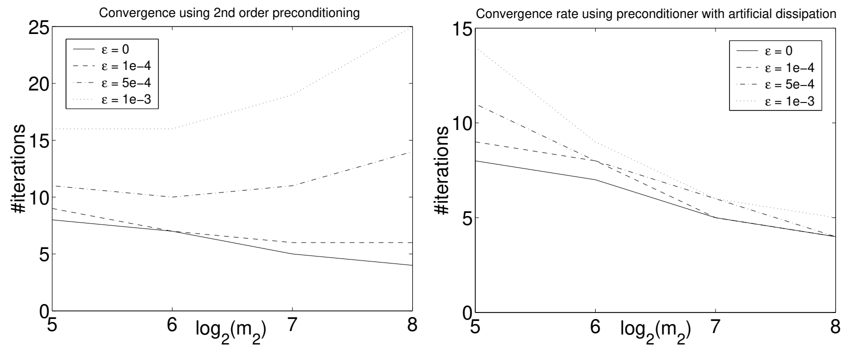

Fig. 7. Plots of the convergence for different amount of artificial dissipation ε. To the left the preconditioner based on second order terms. To the right the artificial dissipation included in the basis of the preconditioner.

We vary the amount of artificial dissipation ε in (3.4) in Fig. 7. It turns out that the preconditioner performs worse when a large amount of artificial dissipation is used. To remedy this behavior we form a different preconditioner that includes the artificial dissipation in the basis. As this gives us a pentadiagonal preconditioner it yields a more costly preconditioner solver. The modified preconditioner works well independently of the amount of artificial dissipation.

grid	no precond.		first order		second order		dissipation	
$m_1 \times m_2$	# it	time(s)	# it	time(s)	# it	time(s)	# it	time(s)
31x31	70	9.0	9	1.5	9	1.5	9	1.5
31x63	56	14.7	9	3.0	7	2.5	8	2.7
31x127	47	25.2	10	6.9	6	4.6	5	3.8
31x255	45	52.0	14	19.4	6	9.2	4	6.6

Table 2. Iteration count and timings for different bases for B in (4.5) when forming the preconditioner.

There is a number of design choices when forming our preconditioner. For instance, it is possible to include only the first order terms in the equation when forming the matrix B in equation (4.5) to obtain M. In Table 2 we compare this approach with the preconditioner based on the full equations with first and second order terms and the modified preconditioner with artificial dissipation. As a reference we also include the numbers for the unpreconditioned solver. The conclusion is that basing M on the first order differences is not optimal for this problem. What we gain in less work forming the preconditioner is lost by its inferior convergence properties compared to the preconditioner based on the full equations with both first and second order differences. However we find that the modified preconditioner performs best of all.

Similar results are obtained for the scalar equation in [12]. The number of iterations decrease for a uniform grid when the number of grid points increase in both directions for different κ (Fig. 7.1) and different quotients m_2/m_1 (Fig. 7.4). The constant $c_h = h_2/\sqrt{\nu}$ which is $0.5-1$ in [12] is here replaced by the relation

$$h_2 \approx 2x_{2,1} = \frac{e^{\eta/m_2}-1}{e^{\eta}-1}$$

from (3.5). For small η we have $h_2 \approx 1/m_2$. The constant $c_v = v/\sqrt{\nu}$ in [12] is not needed here since v in A_2 in (2.6) is determined by the Blasius solution. The number of iterations is almost constant when grid stretching is introduced in the normal direction in Fig. 7.6 in [12] in the same manner as in Fig. 5.

6 Conclusions

We have presented results for preconditioners based on semi-Toeplitz approximations for linearized Navier–Stokes equations that are in good agreement with the theoretical and experimental results for a scalar model problem in [12]. It appears that performance is much improved both in terms of convergence for the iterative solver and in terms of total computation time. The iteration count is improved by a factor seven or more in our experiments, and as the overhead introduced by the preconditioner is rather moderate, the total runtime of the computation is reduced substantially.

A critical feature of a preconditioner is robustness under a variety of conditions. For instance, in the context of fluid computations it is important that the preconditioner works well even when a stretching in the grid is introduced. Our experiments show that this is the case for the semi-Toeplitz preconditioner when we stretch the grid in the x_2-direction. In fact the number of iterations decays the more we stretch, at least for the parameter values investigated here.

It should also be noted that the behaviour of the preconditioner is robust when we increase the size of the time step. This is important in some cases where time resolution is not critical for the choice of time step. Then performance gains can be achieved by increasing the time step for example to reach steady state. From our results we conclude that this is possible for the preconditioned method but not without preconditioning.

In nonlinear problems, solution of a system of linear equations as in (1.1) is part of an outer iteration to solve a nonlinear system of equations. When the initial guess is close to the converged solution in every time step, the iterative behaviour of such problems resembles the behaviour of our linearized equations. In time dependent problems the initial guess is usually the solution from the previous time step or an extrapolation of it. This is often a very good approximation and we can expect the iterative solution process for the compressible Navier–Stokes equations in the boundary layer using our preconditioner to have much in common with the demonstrated efficiency and robustness for the linearized equations.

References

1. Roland W. Freund, Gene H. Golub, and Noël M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1992.
2. Anne Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, Philadelphia, PA, 1997.
3. Lina Hemmingsson. A fast modified sine transform for solving block-tridiagonal systems with Toeplitz blocks. *Numer. Algorithms*, 7:375–389, 1994.
4. Lina Hemmingsson. Toeplitz preconditioners with block structure for first-order PDEs. *Numer. Linear Algebra Appl.*, 3:21–44, 1996.
5. Lina Hemmingsson. A semi-circulant preconditioner for the convection-diffusion equation. *Numer. Math.*, 81:211–248, 1998.

6. Sverker Holmgren, Henrik Brandén, and Erik Sterner. Convergence acceleration for the linearized Navier–Stokes equations using semicirculant approximations. *SIAM J. Sci. Comput.*, 21:1524–1550, 2000.
7. Sverker Holmgren and Kurt Otto. Semicirculant preconditioners for first-order partial differential equations. *SIAM J. Sci. Comput.*, 15:385–407, 1994.
8. Sverker Holmgren and Kurt Otto. Semicirculant solvers and boundary corrections for first-order partial differential equations. *SIAM J. Sci. Comput.*, 17:613–630, 1996.
9. Antony Jameson, Wolfgang Schmidt, and Eli Turkel. Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes. Technical Report AIAA-paper 81-1259, American Institute of Aeronautics and Astronautics, 1981.
10. Arnold M. Kuethe and Chuen-Yen Chow. *Foundations of Aerodynamics: Bases of Aerodynamic Design*. Wiley, New York, 1977.
11. H. Schlichting. *Boundary Layer Theory*. McGraw-Hill, New York, 1979.
12. Samuel Sundberg and Lina von Sydow. Analysis of a semi-Toeplitz preconditioner for a convection-diffusion problem. Technical Report 2002-014, Dept. of Information Technology, Uppsala Univ., Uppsala, Sweden, 2002.