CONVERGENCE ACCELERATION FOR THE STEADY STATE EULER EQUATIONS

HENRIK BRANDÉN* AND SVERKER HOLMGREN*

Abstract. We consider the iterative solution of systems of equations arising from discretizations of the non-linear Euler equations governing compressible flow. The differential equation is discretized on a structured grid, and the steady state solution is computed by a time-marching method.

A convergence acceleration technique based on semicirculant approximations of the difference operator or the Jacobian is used. Implementation issues and variants of the scheme allowing for a reduction of the arithmetic complexity and memory requirement are discussed. The technique can be combined with a variety of iterative solvers, but we focus on non-linear explicit Runge-Kutta time-integration schemes. The results show that the single-stage forward Euler method can be used, and the time step is not limited by a CFL-criterion. Instead it can be chosen as a constant, independent of the number of grid points. This results in that the arithmetic work required for computing the solution is equivalent to the work required for a fixed number of residual evaluations.

Two major advantages of the semicirculant convergence acceleration technique is that it contains few tunable parameters, and that it is robust with respect to the amount of artificial viscosity used in the discretization.

Key words. Convergence acceleration, Semicirculant approximations, Euler equations

AMS subject classifications. 65N22, 65F10, 65M12

1. Introduction. We consider a convergence acceleration technique, based on semicirculant approximations, for the solution of systems of equations arising from systems of non-linear PDE,

$$\mathcal{R}(\mathbf{v}) \equiv \mathcal{A}(\mathbf{v}) - \mathbf{g} = 0. \tag{1.1}$$

Here, $\mathbf{v}$ is the vector of unknown functions, $\mathbf{g}$ is the vector of forcing functions, and $\mathcal{A}$ is a first-order non-linear differential operator given by

$$\mathcal{A}(\mathbf{v}) = \sum_{\nu=1}^{d} \frac{\partial}{\partial x_\nu} \mathbf{f}_\nu(\mathbf{v}), \tag{1.2}$$

or equivalently

$$\mathcal{A}(\mathbf{v}) = \sum_{\nu=1}^{d} A_\nu(\mathbf{v}) \frac{\partial \mathbf{v}}{\partial x_\nu}\ , \quad \text{where} \quad A_\nu \equiv \frac{\partial \mathbf{f}_\nu}{\partial \mathbf{v}}. \tag{1.3}$$

Both forms (1.2) and (1.3) may be used in computations. For example, the conservative form (1.2) is normally used for compressible flow applications where the solution contains shocks, while the non-conservative form (1.3) is often employed for problems describing incompressible flows. Convergence acceleration using semicirculant approximations is applicable for both formulation (1.2) and (1.3).

We assume that the domain and boundary conditions are given such that the PDE problem is well-posed. We also assume that the matrices $A_\nu(\mathbf{v})$ have real eigenvalues, implying that (1.1) is the steady state version of a time-dependent hyperbolic system of PDE,

$$\frac{\partial \mathbf{v}}{\partial t} + \mathcal{R}(\mathbf{v}) = 0. \tag{1.4}$$

*Department of Scientific Computing, Information Technology, Uppsala University, Box 120, S-751 04 Uppsala, Sweden (henrik@tdb.uu.se and sverker@tdb.uu.se)

Problems of this type are encountered in many fields of industry and science. One application problem, studied in some detail later, is the Euler equations for compressible, isentropic flow. The solution schemes described in this paper may also be useful for other problems of wave-propagation or convection type where the steady state solution exists and is of interest. Such problems are found, e.g., in settings where flow- and electromagnetic properties interact.

We consider discretizations on structured grids. For reasons of simplicity, memory requirement, serial efficiency, and parallelization properties, such discretizations are used in many codes in industry. Here, complex geometries are treated using multi-block grids, where the component grids are structured. In Section 6, we present numerical experiments for the non-linear Euler equations in two space dimensions discretized on a single block grid. The extensions to multi-block grids and three space dimension is briefly discussed in Section 7.

2. Iterative methods. When the spatial derivatives in (1.1) are discretized, this results resulting in a large system of non-linear equations for the vector v_∞, representing the steady state solution,

$$R(v_\infty) = 0, \quad \text{or} \quad B(v_\infty)v_\infty = g. \tag{2.1}$$

Here, $R(v)$ denotes the discretized residual, and $B(v)$ is the corresponding non-linear difference operator. Using the same spatial discretization for the time-dependent problem (1.4) results in a large system of non-linear ODE,

$$\frac{dv}{dt} + R(v) = 0\ , \quad \text{or} \quad \frac{dv}{dt} + B(v)v - g = 0. \tag{2.2}$$

We introduce convergence acceleration, represented by a non-singular preconditioning matrix M of the same size as the difference operator B. In practice, neither M nor M^{-1} is normally formed, but an efficient algorithm for the preconditioner solve $w = M^{-1}v$ is supplied.

In general, the matrix M^{-1} could be diagonal or block-diagonal, representing point preconditioners, or narrow block-banded, representing local preconditioners where unknowns at a few neighboring grid points are coupled. Examples of such preconditioners are local time-stepping, preconditioning of the system of PDE, see, e.g., [20], and residual smoothing, see, e.g., [14] [2]. The preconditioner solve may also involve unknowns at many grid points at large relative distances in the grid, corresponding to that M^{-1} is essentially a full matrix. Examples of preconditioners of this type are the semicirculant approximations considered in this paper, and multigrid preconditioners. Both these classes of preconditioners are based on properties of the PDE (1.1). In the semicirculant preconditioner, a modified PDE problem is solved using a fast direct solver, while in a multigrid preconditioner, the original PDE is solved on coarser grids. In both cases, M is formed as an approximation of a difference operator. Then, analysis for model problems and numerical experiments are used to verify that M^{-1} is a good approximation of the inverse of the difference operator, in the sense that the convergence properties for the iterative method is improved.

A standard approach for computing v_∞ is to integrate (2.2) in time until the time derivative of v vanishes. In CFD codes for compressible flow problems, non-linear explicit Runge-Kutta iterations of the following form are often used for the time-marching, [15]

$$
\begin{aligned}
v^{(0)} &= v^i \\
v^{(1)} &= v^{(0)} - \alpha_1 k M^{-1}(v^{(0)}) R(v^{(0)}) \\
&\vdots \\
v^{(s)} &= v^{(0)} - \alpha_s k M^{-1}(v^{(s-1)}) R(v^{(s-1)}) \\
v^{i+1} &= v^{(s)}.
\end{aligned} \tag{2.3}
$$

Here, k denotes the step in (pseudo-)time. Normally, $\alpha_s = 1$ is chosen, resulting in a consistent time-marching method. If $M^{-1} = I$, (2.3) is then a time-accurate iteration.

The constants α_l should be chosen such that $\|R^{n+1}\|$ is reduced by the iteration. The wave propagation property of the PDE implies that the spectrum of $B(v)$ approaches the origin along the imaginary axis. If a preconditioner retaining the wave propagation property is employed, the stability region for the Runge-Kutta method should include a segment of the imaginary axis centered at the origin. A standard scheme, often used in applications, is the three-stage method where $\alpha_1 = \alpha_2 = 0.6$. We will denote this method RK3.

When $M = I$, the CFL stability criterion implies that the time step k has to be of the same order as the smallest space step. For most problems, this leads to that many evaluations of $R(v)$ are required for convergence, resulting in very time-consuming computations. To improve the convergence properties, the preconditioner M is chosen such that the CFL time step criterion is less restrictive or possibly completely removed, resulting in grid independent convergence. The probably most popular convergence acceleration technique for the type of problems of interest here is multigrid, see, e.g., [13] and [19]. Here, grid independent convergence is mainly achieved by retaining the wave propagation property, but increasing the wave propagation speed [18]. However, even-though multigrid techniques often are very successful, such methods contain many parameters and might be difficult to tune. Multigrid methods are also often quite sensitive to the amount and type of artificial viscosity in the discretization, and rely on the possibility of constructing reasonable coarse grid hierarchies. Hence, alternative techniques for convergence acceleration, e.g. based on semicirculant approximations, are worth considering.

If the preconditioner could be chosen such that the wave propagation property is not present in the preconditioned problem, possibly $s < 3$ could be employed without an increase in the number of iterations. Analysis and the numerical experiments presented in Section 6 show that using a semicirculant approximation for preconditioning leads to grid independent convergence. Furthermore, the results presented in Section 6 show that the wave propagation property is removed from the preconditioned problem, and the smallest number of residual evaluations is achieved using $s = 1$, i.e., the forward Euler method.

As the preconditioner M, we will employ a semicirculant approximation of either $B(v)$ or $J(v)$, where is the Jacobian matrix of R. Both choices are motivated by noting that (2.3) is related to two standard solution techniques for the steady state problem (2.1). First, assume that we linearize (2.3) by setting $M^{-1} \equiv M^{-1}(v^i)$ and $R(v) = B(v^i)v - g$. Then, (2.3) is equivalent to solving (2.1) using a preconditioned fixed-point iteration,

$$
M^{-1}(v^i) B(v^i) v^{i+1} = M^{-1}(v^i) g, \tag{2.4}
$$

where the linear systems (2.4) are solved using one step of the linearized version of (2.3) as an inner iteration. Also, assume that we again set $M^{-1} \equiv M^{-1}(v^i)$, but linearize (2.3) by employing $R(v^i + \delta v) = R(v^i) + J(v^i) \cdot \delta v$. Then, (2.3) is equivalent to solving (2.1) using Newton's method, i.e.,

$$v^{i+1} = v^i + \Delta v, \tag{2.5}$$

where the update Δv is computed by solving

$$M^{-1}(v^i) J(v^i) \Delta v = -M^{-1}(v^i) R(v^i), \tag{2.6}$$

again using one step of the linearized version of (2.3) as the inner iteration.

In Section 6, we will present numerical experiments where the type of explicit schemes described above are employed. An alternative, often employed in finite elements settings, is to use an implicit time-integration method. If the backward Euler method is employed for the semi-discrete problem, this results in

$$v^{i+1} = v^i + \Delta v, \tag{2.7}$$

where Δv is computed by solving

$$M^{-1}(v^i) J_A(v^i) \Delta v = -M^{-1}(v^i) R(v^i). \tag{2.8}$$

Here, preconditioning is introduced in (2.8), and $J_A(v^i) \equiv k^{-1} I + J(v^i)$ is an augmented Jacobian. Note that, In the limit $k \to \infty$, this scheme reduces to Newton's method for the steady state problem (2.1). The linear systems (2.8) must normally be solved using an inner iteration. If a stationary Krylov iteration of the type (2.3) is employed, the relation to the explicit schemes described above is clear. Also, a non-stationary scheme like restarted GMRES or mGMRES could be used, in all cases the resulting method is what is normally referred to as a Newton-Krylov iteration. Since the time-marching scheme is implicit, stability does normally not pose a restriction on the time step k. To reduce the total number of residual evaluation, different strategies for choosing varying k are used [17] [12]. Also, if an inner iterative scheme is employed, a method for choosing the convergence criterion for these iterations must be supplied.

3. Semicirculant approximations. In this section we briefly review earlier results for linear steady state problems, providing the basis for the study of non-linear problems presented in this paper.

The semicirculant preconditioner M corresponds to the discretization of a modified linear PDE problem, where the boundary conditions in one spatial direction are altered to periodic, and where the coefficients in the difference operator are constant in this direction [8] [9]. The preconditioning algorithm yields a fast direct solver for the modified PDE problem. Semicirculant preconditioners could also be constructed within a more general framework, based on matrix approximation theory [10] [11].

In [9], a linear hyperbolic model problem is considered, and analysis shows that the number of iterations required is independent of the number of grid points. In [1], the following scalar model problem is considered,

$$\frac{\partial \mathbf{v}}{\partial x_1} + \frac{\partial \mathbf{v}}{\partial x_2} = \mathbf{g}. \tag{3.1}$$

The discretization is performed on a uniform grid on the unit square with n grid points. The derivatives are approximated using standard second-order accurate centered differences, and second order artificial viscosity is added. The corresponding time-dependent problem,

$$\frac{\partial v}{\partial t} + M^{-1}(Bv - g) = 0, \tag{3.2}$$

is preconditioned using the semicirculant approximation of the difference approximation matrix B. The system of ODE (3.2) is integrated using the forward Euler scheme, resulting in

$$v^{i+1} = \left(I - kM^{-1}B\right) v^i + kM^{-1}g.$$

Let λ_ℓ denote the eigenvalues of $M^{-1}B$. A necessary condition for convergence is

$$|1 - k\lambda_\ell| < 1\ , \quad \ell = 1, \ldots, n. \tag{3.3}$$

In [1], the spectrum of $M^{-1}B$ is analyzed, and analytic formulas for λ_ℓ are derived. Using these formulas, it is shown that the time step k is not limited by a CFL-criterion of type $k = \mathcal{O}(h)$, where h is the space step. In fact, $k \approx 1$ can be used. Furthermore, analysis shows that, if the grid has more grid points in the direction where constant coefficients and periodic boundary conditions are used in the preconditioner, the artificial viscosity may be arbitrarily small. Also, some numerical experiments are presented for the linearized Euler equations, showing that the same type of convergence properties is also achieved for this system of linear hyperbolic PDE.

In [7], the linearized Navier-Stokes equations for steady state compressible flow over a flat plate is considered, and a model problem is analyzed. Here, the results show that the convergence rate is independent of the space step and the viscosity parameter. In [4], the scalar convection-diffusion equation is considered, and in [5], the fast convection-diffusion solver is employed in a scheme for the incompressible Navier-Stokes equations.

4. The non-linear Euler equations. We consider the steady-state Euler equations

$$A_1(\mathbf{v})\frac{\partial \mathbf{v}}{\partial x_1} + A_2(\mathbf{v})\frac{\partial \mathbf{v}}{\partial x_2} = 0, \tag{4.1}$$

where $\mathbf{v} = (\rho, v_1, v_2)^T$, ρ is the density and v_1 and v_2 are the velocity components. We assume isentropic flow, implying that the pressure p is given by $p = \beta\rho^\gamma$. Here, we employ $\beta = 2$ and $\gamma = 1.4$. The speed of sound is $c = \sqrt{\gamma\beta\rho^{\gamma-1}}$.

The coefficient matrices $A_{1,2}$ are given by

$$A_1(\mathbf{v}) = \begin{pmatrix} v_1 & \rho & 0 \\ c^2/\rho & v_1 & 0 \\ 0 & 0 & v_1 \end{pmatrix}$$

and

$$A_2(\mathbf{v}) = \begin{pmatrix} v_2 & 0 & \rho \\ 0 & v_2 & 0 \\ c^2/\rho & 0 & v_2 \end{pmatrix}.$$

We consider sub-sonic flow in a non-axisymmetric duct, see Figure 4.1. Here, the boundary conditions are given by prescribing the in-going characteristics at the in-and outflow boundaries, and setting the flow perpendicular to the walls equal to zero, i.e.,

$$\begin{array}{lll} & \rho + v_1 = g_1 \text{ and } v_2 = g_2 & \text{at inflow boundary,} \\ (4.2) & \rho - v_1 = g_3 & \text{at outflow boundary,} \\ & \hat{n} \cdot (v_1, v_2) = 0 & \text{at walls.} \end{array}$$

In the experiments we use $g_1 = 2$, $g_2 = 0$ and $g_3 = 0$.

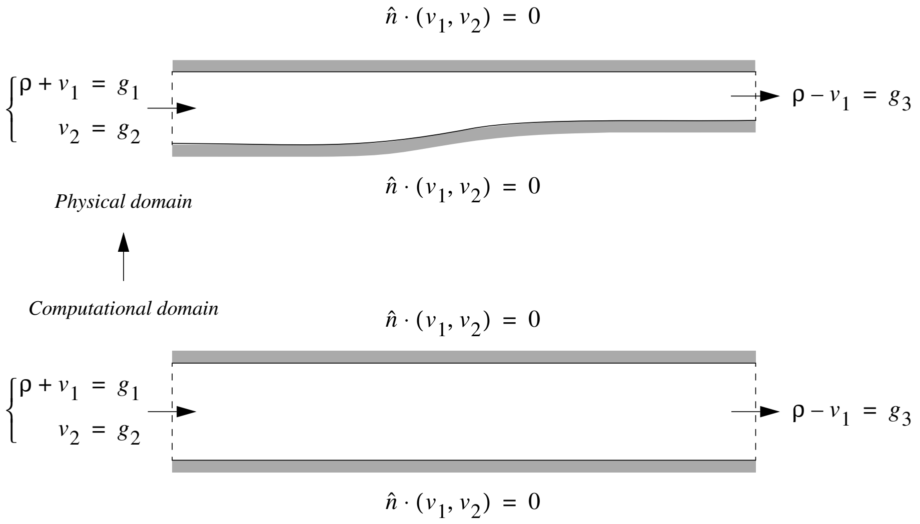

FIG. 4.1. *The physical and the computational domain.*

The discretization is performed using centered differences on an orthogonal grid, which simplifies the implementation. However, note that the semicirculant preconditioning technique is not limited to this situation. The discretization procedure is as follows:

1. An orthogonal mapping is defined from the physical domain to the rectangular computational domain, see Figure 4.1.
2. The equations are transformed from the physical to the computational domain.
3. The transformed equations are discretized on an $m_1 \times m_2$ uniform grid in the computational domain. The derivatives are approximated using second order centered finite differences, and fourth-order artificial viscosity is added. For application problems, a standard value for the artificial viscosity parameter is $\gamma_4 = 1/32$. For accuracy reasons, a smaller value for γ_4 would be preferred, but this often leads to instability caused by oscillatory parasitic solutions.
4. The boundary conditions (4.2) are used for the in-going characteristics, and the outgoing characteristics are extrapolated. At walls, the tangential velocity as well as the density are extrapolated. The boundary conditions are used to eliminate the equations corresponding to boundary points.

5. Implementation. The computations involved in using the semicirculant preconditioner can be separated into three phases, corresponding to three top-level routines which are called from the iterative solver:

1. Preconditioner construction. Computes the parameters describing the semicirculant preconditioner M.

2. Preconditioner factorization. Computes the LU-factors for the small systems of equations in the factorization of M^{-1}, see below.
3. Preconditioner solve. Computes $w = M^{-1}v$ using the factorization determined in phase 2.

Note that if the same preconditioner M is reused for more than one residual evaluation, phases 1 and 2 only have to be performed when M is modified.

The algorithms for the different phases, including parallelization strategies, are described in detail in [6]. Below, we discuss implementation aspects in the context of using the preconditioner in a code for solving a non-linear system of n_c PDE on an $m_1 \times m_2$ grid, exemplified by the discretization of the Euler equations described in Section 4. For comparison, one evaluation of the residual $R(v)$ requires approximately 120 arithmetic operations per grid point for this problem.

Preconditioner construction: The preconditioner M is defined by a difference approximation stencil with periodic boundary conditions in the x_1-direction and constant coefficients in the same direction. If there are n_z non-zero coefficients in the stencil, M is determined by $n_z m_2$ parameters. If these parameters are computed by averaging the coefficients in the stencil corresponding to a difference approximation matrix A, the resulting approximation M minimizes $\|M - A\|_F$ [10]. For the averaging, at most n_z arithmetic operations per grid point are required.

A straight-forward way of constructing M is by computing the averages in the x_1-direction for the stencils corresponding to B or J. However, it is also possible to base the construction of M on a modified stencil. For example, let B_2 correspond to a discretization of the Euler equations where second order artificial viscosity replaces the fourth order term used in B. Then, the difference stencil corresponding to B_2 will be smaller than the one corresponding to B, see Figure 5.1. This will result in that $n_z = 31$, compared to $n_z = 43$ for B. Also, the smaller stencil for B_2 will result in substantial reductions of the arithmetic complexities and memory requirements in the preconditioner factorization and solve, see below.

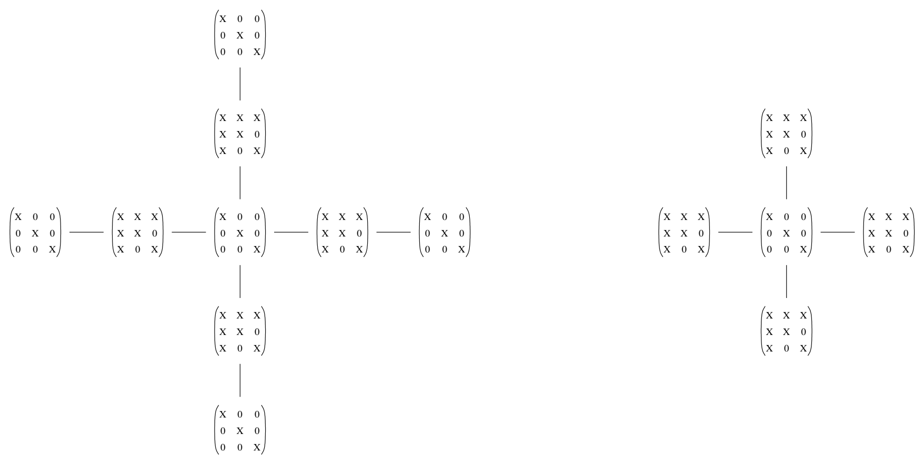

FIG. 5.1. *The stencils corresponding to B (left) and B_2 (right) for the Euler equations.*

It should be noted that in existing CFD codes using explicit time-marching, often only a routine for computing $R(v)$ is supplied. In codes using an implicit time-marching scheme, a matrix-free implementation is often employed. Here, the action of the Jacobian is computed using numerical differentiation,

$$J(v)w \approx \frac{R(v+\sigma w) - R(v)}{\sigma}.$$

Hence, the entries of $B(v)$ or $J(v)$ may not be explicitly available, and some programming effort might be required for determining the averages required for constructing M.

Preconditioner factorization: The basis for the fast preconditioner solve is a factorization of the form [8]

$$M^{-1} = Q\Lambda^{-1}Q^{*}. \tag{5.1}$$

Here, Q and Q^* correspond to $n_c m_2$ independent discrete Fourier transforms, and Λ corresponds to $m_1/2+1$ independent narrow-block-banded systems with block size n_c and $n_c m_2$ unknowns. As remarked above, it is normally advantageous to factorize these systems once and store the factors. For the Euler equations, the systems are block-pentadiagonal if B or J is used for constructing M, and block-tridiagonal if B_2 or J_2 is employed. Since $n_c = 3$, the block size is 3×3. In Table 5.1, the arithmetic complexity for the preconditioner factorization and memory requirement for storing the factors are given.

TABLE 5.1
Arithmetic complexity and memory requirement for preconditioner factorization for the Euler equations

M constructed from	a.o. per grid point	memory positions per grid point
B or J	768	54
B_2 or J_2	300	27

Preconditioner solve: Employing the factorization (5.1), the preconditioner solve is performed in three steps. For the Euler equations, these consists of

1. $3m_2$ independent FFTs of real data, length m_1.
2. Back-substitution for $m_1/2+1$ independent block-pentadiagonal or block-tridiagonal systems of equations with 3×3 blocks and $3m_2$ unknowns.
3. $3m_2$ independent inverse FFTs of conjugate even data, length m_1.

In Table 5.2, the arithmetic complexity for the preconditioner solve is given.

TABLE 5.2
Arithmetic complexity for preconditioner solve for the Euler equations

M constructed from	a.o. per grid point
B or J	$15\log_2 m_1 + 312$
B_2 or J_2	$15\log_2 m_1 + 204$

6. Numerical experiments. We present experiments where we solve the non-linear Euler equations in the non-axisymmetric duct described in Section 4, using the type of explicit time-marching schemes described in Section 2. We have chosen to focus on explicit schemes, but we have also employed the semicirculant preconditioning technique in the type of Newton-Krylov iterations based on implicit time-discretizations. As remarked in Section 2, the two types of schemes are closely related, and in general the convergence results are very similar.

If not otherwise stated, the initial guess is given by $\rho \equiv 1$, $v_1 \equiv 1$, and $v_2 \equiv 0$. We terminate the iterations when the residual has been reduced by a factor of 10^5. In Figure 6.1, ρ, v_1, v_2 and c for the converged solution v_∞ are shown. The Mach number at the inflow is approximately 0.57.

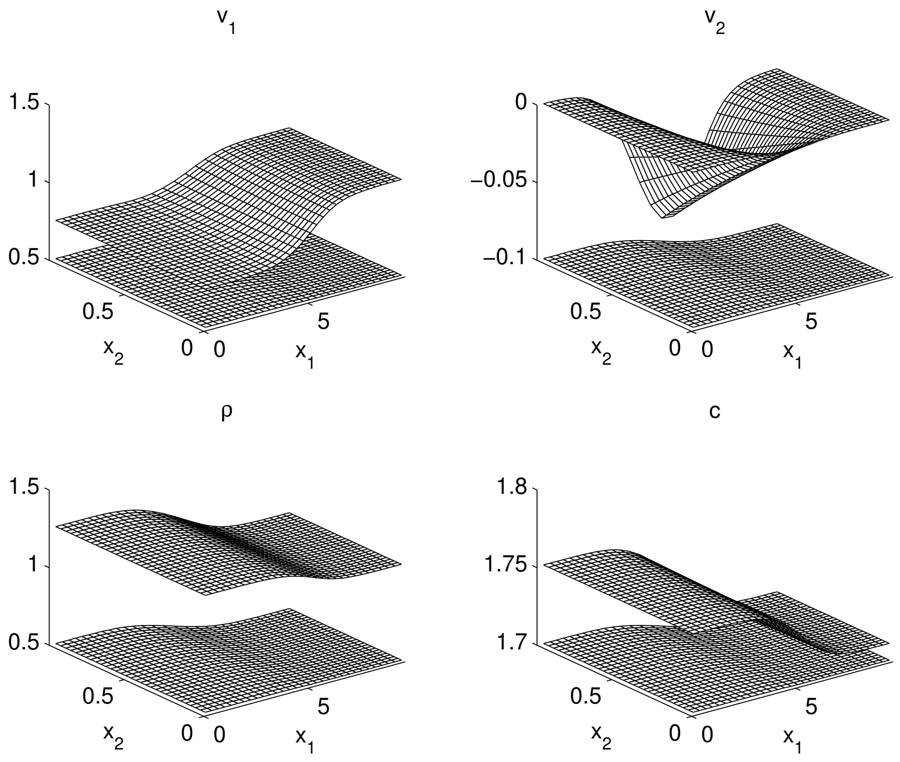

FIG. 6.1. *The stationary flow in a non-axisymmetric duct governed by the non-linear Euler equations.*

In the first experiment, we study iterations of the type (2.3) where a semicirculant approximations of J is employed for preconditioning. The artificial viscosity parameter is chosen as $\gamma_4 = 1/32$. In Table 6.1, the number of evaluations of $M^{-1}R$ required for convergence when using the forward Euler method is compared to the results for the three-stage RK3 scheme. Experiments show that for the preconditioned forward Euler method, $k = 0.4$ yields the best performance for all problem sizes. For the preconditioned RK3 method, the same type of experiments show that $k = 0.6$ should be used.

TABLE 6.1
A comparison between the RK3 and the forward Euler schemes

$m_{1,2}$	8	16	32	64
Evaluations of $M^{-1}R$, forward Euler method	48	51	54	54
Evaluations of $M^{-1}R$, RK3 method	93	90	90	90

The results in Table 6.1 show that the number of evaluations of the preconditioned residual is almost constant for both iterative methods. Also, using the single stage forward Euler scheme is more efficient than employing the standard RK3 method. In the sequel, the forward Euler method with $k = 0.4$ is used if not otherwise is stated.

In Figure 6.2, the spectrum of the iteration matrix $I - 0.4 \cdot M^{-1}(v_\infty)B(v_\infty)$ is shown for two different grid sizes. The solid line shows the stability region for the forward Euler iteration. The results in Table 6.1 and Figure 6.2 show that the time

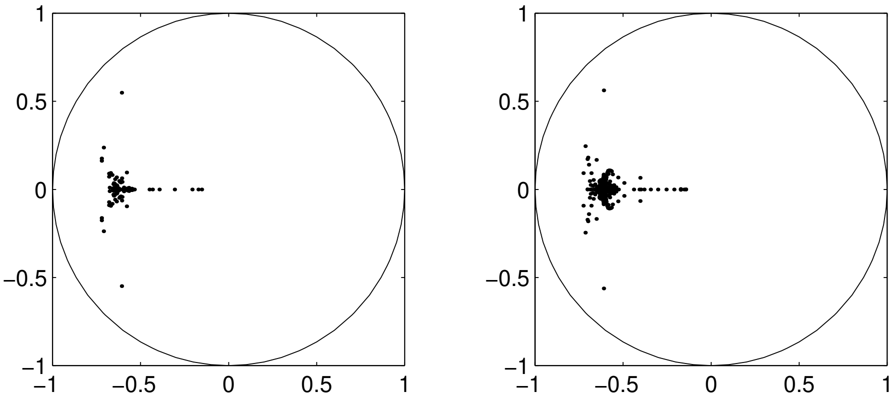

FIG. 6.2. *Spectra of the forward Euler iteration matrix for* $m_{1,2} = 8$ *and* $m_{1,2} = 16$.

step k is not limited by a CFL-like criterion.

In Figure 6.3, the evolution of the density ρ is shown for a time-accurate iteration. Here, the initial guess is given by a pulse close to the inflow. In the figure, the wave propagation property of the problem is clearly seen. The pulse is propagated in the flow direction, while sound waves are reflected back and forth between the walls.

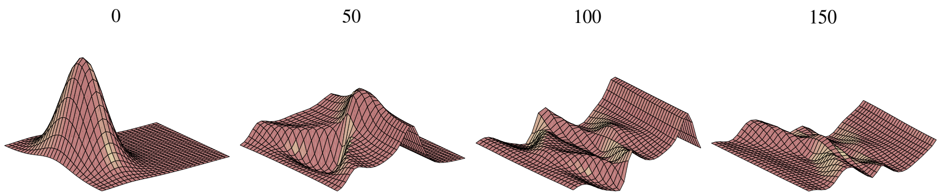

FIG. 6.3. *The density* $\tilde{\rho}$ *computed using the time-accurate RK3 method. The pictures show the state after 0, 50, 100 and 150 iterations respectively.*

In Figure 6.4, the same initial guess is chosen, but here the preconditioned forward Euler method is employed. The figure shows that there are basically no wave propagation properties apparent in the preconditioned problem. The pulse does not move, instead it is damped by the iteration.

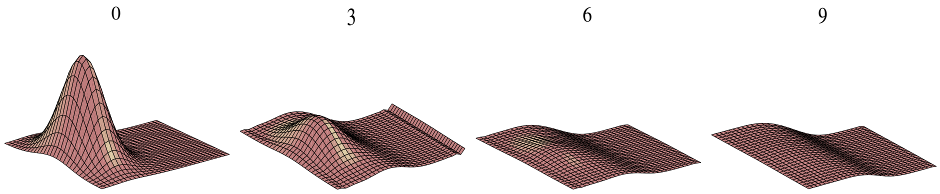

FIG. 6.4. *The density $\tilde{\rho}$ computed using the preconditioned forward Euler method. The pictures show the state after 0, 3, 6 and 9 iterations respectively.*

Next, we in Tables 6.2, 6.3, and 6.4 compare using the semicirculant approximation of B and J for preconditioning. Again, we employ fourth order artificial viscosity, and examine the behavior for different values of γ_4.

TABLE 6.2
Comparison of $M \approx B$ and $M \approx J$, $\gamma_4 = 1/32$

$m_{1,2}$	8	16	32	64
Evaluations of $M^{-1}R$, $M \approx B$	52	56	57	57
Evaluations of $M^{-1}R$, $M \approx J$	48	51	54	54

TABLE 6.3
Comparison of $M \approx B$ and $M \approx J$, $\gamma_4 = 1/1024$

$m_{1,2}$	8	16	32	64
Evaluations of $M^{-1}R$, $M \approx B$	48	55	64	64
Evaluations of $M^{-1}R$, $M \approx J$	41	46	59	64

TABLE 6.4
Comparison of $M \approx B$ and $M \approx J$, $\gamma_4 = 0$

$m_{1,2}$	8	16	32	64
Evaluations of $M^{-1}R$, $M \approx B$	58	56	55	55
Evaluations of $M^{-1}R$, $M \approx J$	div	div	div	div

The results in Tables 6.2 and 6.3 show that, for $\gamma_4 = 1/32$ and $\gamma_4 = 1/1024$, using $M \approx J$ yields slightly better performance than employing $M \approx B$. Also, it is clear that the convergence rate is almost independent of γ_4. As seen in Table 6.4, utilizing $M \approx B$ results in a convergent iteration even for $\gamma_4 = 0$. In fact, the convergence is slightly faster than for $\gamma_4 > 0$. For $\gamma_4 = 0$, the method using $M \approx J$ fails to converge.

Now we investigate the convergence properties when employing $M \approx J_2$ instead of $M \approx J$, hereby reducing the arithmetic work and memory requirement. Here, the parameter for the second order artificial viscosity used in J_2 is chosen as $\gamma_2 = \gamma_4$. In Table 6.5, results for $\gamma_{2,4} = 1/32$ and $\gamma_{2,4} = 1/1024$ are shown.

TABLE 6.5
Performance for $M \approx J_2$

$m_{1,2}$	8	16	32	64
Evaluations of $M^{-1}R$, $\gamma_{2,4} = 1/32$	48	44	46	46
Evaluations of $M^{-1}R$, $\gamma_{2,4} = 1/1024$	48	46	52	62

By comparing the results in Table 6.5 to those in Tables 6.2 and 6.3, we see that for large problems, the number of evaluations of the preconditioned residual is actually reduced when $M \approx J_2$ is used instead of $M \approx J$.

Finally, we instead of constructing and factorizing M in every iteration, introduce an ad-hoc strategy for determining when to rebuild the preconditioner. Let $\rho^i \equiv \|R(v^i)\|/\|R(v^0)\|$. If $\rho^i > 1.0$, M is rebuilt in every iteration. If $0.1 < \rho^i < 1.0$, M is rebuilt in every fifth iteration, and if $0.01 < \rho^i < 0.1$ M is rebuilt in every 10:th iteration. Finally, if $\rho^i < 0.01$, M is not rebuilt anymore. Introducing this strategy and repeating all of the experiments in Table 6.5, we find that the number of evaluations of $M^{-1}R$ is not increased by more than at most one, while the number of preconditioner rebuilds are reduced to five or six.

In Figure 6.5, we compare the arithmetic work for the unpreconditioned RK3 method to the preconditioned forward Euler scheme, where $M \approx J_2$ and the preconditioner rebuild strategy described above is employed. Here, $\gamma_{2,4} = 1/32$, and for the unpreconditioned method, the time step is given by the CFL criterion,

$$k = \frac{1.5}{\dfrac{v_{1,max}}{h_{1,min}} + \dfrac{v_{2,max}}{h_{2,min}} + c_{max}\sqrt{h_{1,max}^{-2} + h_{2,max}^{-2}}}.$$

Here, $h_{d,min}$ denotes the minimal space step in the d-direction etc. The arithmetic work is given in residual evaluation equivalents, i.e., it is normalized by the number of arithmetic operations required for evaluating $R(v)$. The dashed line corresponds to the RK3 method, and the solid line represents the preconditioned forward Euler scheme. From Figure 6.5, it is clear that the preconditioned forward Euler iteration is

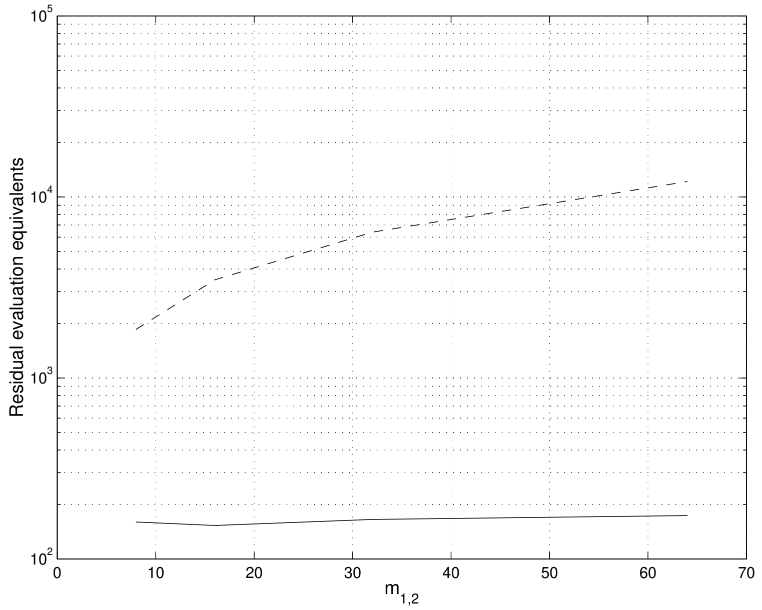

FIG. 6.5. *Comparison of arithmetic work for the standard, unpreconditioned Runge-Kutta method and the forward Euler method employing $M \approx \tilde{J}$ and the ad-hoc rebuild strategy.*

a clear winner for all problem sizes. Also, the number of residual evaluation equivalents required for computing a converged solution does not grow when the number of grid points is increased. For $\gamma_4 = 1/1024$, the convergence for the unpreconditioned method is so slow that it is not practical to compute a converged solution for large number of grid points. Hence, a comparison of the type presented in Figure 6.5 is not possible for this case.

7. Concluding remarks. In [9] and [1], analysis and experiments for linear, scalar, steady state problems show that the semicirculant convergence acceleration technique has favorable convergence properties. The numerical experiments presented in Section 6 show that these properties are retained also for an example application, where the non-linear Euler equations are solved in a non-axisymmetric duct. The main result is that the time step k is not limited by a CFL-criterion. On the contrary, k can be chosen independent of the space step, and experiments show that the work per grid point required for computing a converged solution is bounded by a constant, independent of the number of grid points.

The experiments also show that, when using the semicirculant preconditioning scheme, the convergence rate is very insensitive to the amount and type of artificial viscosity added to the discretization. For the flow problem studied in Section 6, it is in fact possible to compute a solution for zero artificial viscosity in an efficient way.

The analysis and the numerical experiments described above show that semicirculant preconditioning yields efficient solution methods for some problems. To determine if this also holds for realistic application problems, further experiments for more complex flow situations and geometries must be performed.

In the problem studied in Section 6, the main direction of the flow coincides with the direction in which the coefficients are averaged when constructing the preconditioner. If the problem does not have a main flow direction, e.g., it represents a situation with circulating flow, it is probable that the technique will not be very successful in the form described above. However, if the domain is divided into a number of component grids forming a multi-block grid for the whole problem, the situation may be different. Possibly, there will now be a main flow direction for each grid, and semicirculant preconditioning could be used in a domain-decomposition framework. In [3] and [16], a similar preconditioner is successfully used in this context.

The extension of the semicirculant preconditioning technique to problems in three space dimensions is straightforward. In some situations, it may be possible get an efficient preconditioner by performing the averaging in two space directions, resulting in a preconditioner solve which is very similar to the one described in Section 5, but where independent 2D FFTs are performed instead of one-dimensional transforms. If the averaging procedure can be applied only in one direction, the systems of equations that have to be solved in phase 2 of the preconditioner solve will correspond to problems in two space dimensions. Even though these systems will be much smaller than the original system, it is not clear if it will be possible to solve them efficiently enough for the preconditioning technique to be competitive.

REFERENCES

[1] H. Brandén and S. Holmgren, *Convergence acceleration for hyperbolic systems using semicirculant approximations*, J. Sci. Comput., 14 (1999), pp. 357–393.

[2] R. Enander, *Implicit explicit residual smoothing for the multidimensional Euler and Navier–Stokes equations*, SIAM J. Sci. Comput., 18 (1997), pp. 1243–1254.

[3] L. Hemmingsson, *Domain Decomposition Methods and Fast Solvers for First-order PDEs*, Ph.D. thesis, Dept. of Scientific Computing, Uppsala Univ., Uppsala, Sweden, 1995.

[4] ———, *A semi-circulant preconditioner for the convection-diffusion equation*, Numer. Math., 81 (1998), pp. 211–248.

[5] L. Hemmingsson-Frändén and A. Wathen, *A nearly optimal preconditioner for the Navier–Stokes equations*, Report NA-99/15, Oxford Univ. Computing Lab., Oxford Univ., Oxford, UK, 1999.

[6] S. Holmgren, *CG-like iterative methods and semicirculant preconditioners on vector and parallel computers*, Report 148, Dept. of Scientific Computing, Uppsala Univ., Uppsala, Sweden, 1992.

[7] S. Holmgren, H. Brandén, and E. Sterner, *Convergence acceleration for the linearized Navier–Stokes equations using semicirculant approximations*, SIAM J. Sci. Comput., 21 (2000), pp. 1524–1550.

[8] S. Holmgren and K. Otto, *Iterative solution methods and preconditioners for block-tridiagonal systems of equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 863–886.

[9] ———, *Semicirculant preconditioners for first-order partial differential equations*, SIAM J. Sci. Comput., 15 (1994), pp. 385–407.

[10] ———, *A framework for polynomial preconditioners based on fast transforms I: Theory*, BIT, 38 (1998), pp. 544–559.

[11] ———, *A framework for polynomial preconditioners based on fast transforms II: PDE applications*, BIT, 38 (1998), pp. 721–736.

[12] E. Issman, G. Degrez, and H. Deconinck, *Implicit upwind residual-distribution Euler and Navier–Stokes solver on unstructured meshes*, AIAA J., 34 (1996), pp. 2021–2028.

[13] A. Jameson, *Solution of the Euler equations for two-dimensional transonic flow by a multigrid method*, Appl. Math. Comput., 327-355 (1983).

[14] A. Jameson, *Computational transonics*, Comm. Pure Appl. Math., 41 (1988), pp. 507–549.

[15] A. Jameson, W. Schmidt, and E. Turkel, *Numerical solution of the Euler equations by the finite volume method using Runge-Kutta time-stepping schemes*, AIAA, 81-129 (1981).

[16] A. Kähäri and S. Sundberg, *Convergence acceleration for the Euler equations using a parallel semi-Toeplitz preconditioner*, in Euro-Par'99: Parallel Processing, P. Amestoy, P. Berger, M. Daydé, I. Duff, V. Frayssé, L. Giraud, and D. Ruiz, eds., vol. 1685 of Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, 1999, pp. 1124–1127.

[17] C. T. Kelly and D. E. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM J. Numer. Anal., 35 (1998), pp. 508–523.

[18] P. Lötstedt, *Grid independent convergence of the multigrid method for first-order equations*, SIAM J. Numer. Anal., 29 (1992), pp. 1370–1394.

[19] ———, *Improved convergence to the steady state of the Euler equations by enhanced wave propagation*, J. Comput. Phys., 114 (1994), pp. 34–44.

[20] E. Turkel, R. Radespiel, and N. Kroll, *Assessment of preconditioning methods for multidimensional aerodynamics*, Comput. & Fluids, 26 (1997), pp. 613–634.