Journal of Engineering Mathematics **38:** 403–426, 2000.

Application of polynomial preconditioners to conservation laws

BERNARD J. GEURTS[1,*], RENÉ VAN BUUREN[1] and HAO LU[2]
[1]*Faculty of Mathematical Sciences, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*
e-mail:b.j.geurts@math.utwente.nl
[2]*Institute for Scientific Computation, Texas A&M University, College Station, TX 77843-3404, U.S.A.*

Received 22 January 1999; accepted in revised form 28 March 2000

Abstract. Polynomial preconditioners which are suitable in implicit time-stepping methods for conservation laws are reviewed and analyzed. The preconditioners considered are either based on a truncation of a Neumann series or on Chebyshev polynomials for the inverse of the system-matrix. The latter class of preconditioner is optimal in a space of polynomials of certain degree if the matrix has only real eigenvalues and a non-singular system of eigenvectors. The preconditioning can be applied to any convergent splitting of the system matrix, *i.e.* to any classical implicit time-stepping method for conservation laws that is based on a quasi-Newton iteration. An efficient implementation based on SSOR is presented and the approach is applied to simulations of the viscous unsteady Burgers equation and to inviscid steady flow around an airfoil in two spatial dimensions to illustrate the method in large-scale computations. For viscous flows the efficiency increase due to preconditioning is considerable.

Key words: polynomial preconditioning, conservation laws, implicit timestepping, aerodynamic flows

1. Introduction

Many problems of mathematical physics can be formulated in terms of conservation laws of the general form

$$u_t + \nabla \cdot (F(u, \nabla u)) = 0, \tag{1}$$

where $u = (u_1, \ldots, u_s)^T$ and each component u_j is a function of time t and spatial coordinates $x_1, x_2, \ldots, x_N$, and F is a (nonlinear) differential operator, which contains derivatives with respect to the spatial coordinates. An important motivating example in this paper is the system of Navier-Stokes equations for compressible flow which can be cast in this form. The efficient time-integration of unsteady solutions to (1) is of central importance in many applications, such as direct and large-eddy simulation of transitional and turbulent flow [1]. Moreover, within pseudo-time relaxation methods [2] to obtain steady solutions to (1) this aspect is equally relevant. We will focus on possible efficiency enhancement arising from the application of polynomial preconditioners, both in steady and unsteady problems.

In a great number of technological and natural systems the flow of fluids plays an important role. A detailed knowledge of the dominant mechanisms which determine the main features of the flow is often essential in order to understand the functioning of such systems and to be able to optimize specific aspects. The physical basis of the governing 'Navier–Stokes' equations is quite simple and represents the conservation of mass, momentum and energy. The nonlinear nature of these equations, however, seriously complicates a quantitative mathematical analysis

*Also: Department of Engineering, Queen Mary and Westfield College, University of London, Mile End Road, London EJ 4NS, U.K.

and an important alternative for investigating this system of equations is formed by numerical simulation. In this context either the Navier-Stokes equations or a filtered and modelled 'physical equivalent' such as the Reynolds averaged Navier-Stokes or Large-Eddy equations serve as point of departure for the numerical treatment. Presently, steady calculations can be performed within an acceptable computational effort, for complex situations involving flows of direct practical interest. Complex, unsteady phenomena such as associated with turbulent flow, however, can not be treated in such generality with sufficient accuracy using present-day computing facilities and numerical methods. Of prime concern in relation to (1) is the development of efficient time-integration schemes which allow to simulate the evolution of the solution u with time. There are many explicit time-integration schemes for conservation laws (see LeVeque [3, 4] for a survey). All explicit methods encounter a time-step constraint due to stability requirements. This constraint can become much too severe compared to restrictions on the time-step arising from temporal accuracy considerations. This is observed in many practically relevant cases of time-dependent fluid flow, *e.g.* transitional and turbulent flow in shear layers [5–7]. Moreover, if only a steady-state solution is desired, this stability constraint limits the time step to unnecessarily small values, in particular if the computation is near convergence.

To relax the numerical constraint on the size of the time-step we consider A-stable implicit time-stepping methods. Within this class of methods, advancing the solution one time-step requires to solve a very large system of coupled algebraic equations. In order to realize this solution many possible methods have been proposed which are all 'inspired' to some degree by the classical Newton method. In view of computational efficiency it is often expedient to use a method that requires only approximate knowledge of the Jacobi matrix of the system of equations. These 'quasi-Newton' methods are aimed at providing a good balance between convergence-rate and computational effort. The problem of solving a linear system of equations is at the heart of all these approaches. In this context preconditioning methods can greatly enhance the efficiency of the overall method [8] and in this paper we turn to a specific class of preconditioners which are suited for conservation laws.

The polynomial preconditioning considered in this paper has the benefit that it can be applied to both viscous and inviscid unsteady flows which are treated with an implicit time-integration method. Moreover, the preconditioning can be beneficial for steady flow calculations as well if a pseudo-time-stepping method is used as relaxation method and the pseudo-time evolution is treated with an implicit method. Here we illustrate the preconditioning for unsteady viscous flow in two dimensions and steady inviscid flow around an airfoil. Extension to viscous unsteady flow governed by the full Navier-Stokes equations, *e.g.*, when simulating turbulent flow phenomena with direct or large-eddy simulation does not give rise to any principal problems. At sufficient resolution, *i.e.*, sufficiently low cell-Reynolds numbers, the viscous contributions to the dynamics are beneficial for the preconditioning efficiency. The computational overhead associated with the preconditioning is quite limited and in two and three dimensions involves only a relatively small number of additional matrix-vector multiplications which amounts to about 5–10% added operations in typical cases.

The actual convergence properties of the (quasi-) Newton iteration method are reflected by spectral and eigenspace properties of the Jacobi matrix of the system of coupled algebraic equations which we denote by A. For a real nonsingular matrix A a polynomial preconditioner of A is a polynomial approximation $p(A)$ of the inverse A^{-1}. Polynomial preconditioning has been used earlier for symmetric positive definite linear systems. In 1979 , Dubois, Greenbaum and Rodrigue [11] proposed using a truncated Neumann series as an approximation of A^{-1} for

the purpose of preconditioning conjugate gradient methods to solve linear systems $A\mathbf{x} = \mathbf{b}$ in combination with parallel computation. Four years later, Johnson, Micchelli and Paul [12] proposed a polynomial preconditioner based on Chebyshev polynomials, which minimizes an upper bound of the condition number of a preconditioned matrix for symmetric positive definite matrices. As a basic preconditioning technique polynomial approximation is used also in the construction of other preconditioners, for example, block incomplete factorization [13], compensative block incomplete factorization [14] and multilevel methods [15]. Compared with other preconditioning methods polynomial preconditioning has two prominent advantages: (a) it needs very limited additional computation to construct a preconditioner and (b) it is easily implemented for parallel computation. These two elements can lead to a more efficient method for time-dependent flow simulation, in particular, compared to explicit time-stepping methods. Some important work on polynomial preconditioning addressing also several practical issues can also be found in [11, 16]. For the simulation of unsteady solutions the fact that combinations of the solution at previous time-levels can be used in an extrapolation to estimate the next solution with great accuracy is very helpful. In particular, this reduces the number of iterations that is required to only a few per time-step and favors quasi-Newton methods.

In this paper we consider polynomial preconditioners arising from the simple truncated Neumann series or from Chebyshev polynomials. The latter can be shown to be 'optimal' in the space of polynomials of specific degree provided the eigenvectors of the system-matrix are not too close to linear dependence and only simple information of the eigenvalue spectrum is incorporated. Alternative polynomials generated by GMRES or BiCGstab have been studied as well and were found to perform slightly less well for the problems considered in this paper. For unsteady flow in two dimensions governed by the viscous Burgers equation, both GMRES and BiCGstab operate about 25% less efficiently compared to the Chebyshev preconditioning but are more effective than the preconditioner arising from truncation of the Neumann series. Actually, for linear systems it can be shown that the truncated Neumann series in combination with a convergent splitting is equivalent to using the convergent splitting alone [9]. The benefit of this 'Neumann' polynomial preconditioner for actual evolution problems based on basic equations such as the viscous Burgers equation or the Navier–Stokes equations appears to be associated with the convergence of the 'outer' iteration process, *i.e.*, with the nonlinearity of the problem. This is not true for the Chebyshev preconditioner or more traditional preconditioners such as GMRES or BiCGstab. The latter preconditioners can show an enhanced efficiency for linear as well as nonlinear problems and can enhance the overall efficiency of a simulation without adding much computational overhead. The use of more advanced Krylov methods [16] can lead to a strong increase of memory usage which for our main long-term goals of direct and large-eddy simulation of turbulent flow, can be a major shortcoming of these methods. Moreover, a stronger benefit for Krylov methods has been reported for steady flow problems [38]; the requirements for unsteady flow simulations are quite different and provide a valuable area of application of polynomial preconditioners. A good discussion of advantages and disadvantages of polynomial preconditioning can be found in [10]. Among the advantages are its relative simplicity in actual implementations which require only matrix-vector products, its reduction of the number of operations, *e.g.*, combined with conjugate gradient methods and the fact that it can be added to enhance an already existing relaxation method. Polynomial preconditioners can be very useful in combination with certain specific complex linear problems. A main disadvantage of polynomial preconditioning is its relatively poor performance on parallel computers with few processors and the fact that for certain

classes of problems good alternatives exist that may perform better. Operating in the 'middle-ground' between no preconditioning and advanced Krylov methods is the main motivation of polynomial preconditioning since we try to find a proper balance between on the one hand the computational overhead that arises from the application of a preconditioner and on the other hand the gain in computational speed that results. In particular, we investigate preconditioners which are suited for time-dependent problems; this does not necessarily imply to use the 'best' preconditioner in the sense of best approximation of the inverse of the system matrix but rather to aim for an optimal computation-time.

Preconditioning methods to speed-up flow simulations have been considered from various perspectives. As a general characteristic, preconditioning that reckons more with specific problem-dependent information, is likely to perform better than methods that try to operate without such input. Moreover, preconditioning itself represents some computational overhead which should clearly be small compared to the associated gain in overall computational efficiency of the simulation. In this paper we consider polynomial preconditioning which incorporates general information concerning the eigenvalue spectrum of the system matrix. This is shown to be beneficial for the efficiency of the preconditioner, *e.g.*, compared to methods which do not involve such problem specific input. A further step in this context was pointed out by Darmofal and Schmid [17]. The importance of eigenvectors in effective preconditioner design was considered by these authors. A lack of eigenvector orthogonality was shown to lead to a strong sensitivity of the simulation method and can even lead to an unrealistic short-time growth of perturbations. In problems which involve system-matrices with 'near-dependent' eigenvectors, effective preconditioning design requires more problem-specific input than just properties of the eigenvalue spectrum. In particular, severe robustness problems were reported to arise around stagnation points and preconditioning would require further problem-specific input. In this paper we do not incorporate eigenvector information and only consider efficiency enhancement arising from incorporating eigenvalue spectrum information. This is shown to be applicable to flows with stagnation points as well. Further improvements for such flows with more involved preconditioners is likely as far as convergence rate is concerned. However, this can also become problem-dependent since in general a better preconditioner will also represent a larger computational overhead. Thus a proper balance between enhancing efficiency and computational overhead and general applicability of the approach must be maintained. Here, for two applications we use only eigenvalue information and develop preconditioning for time-dependent problems. Darmofal and Schmid focused mainly on steady flow at low Mach numbers and modified a preconditioner of Van Leer and Turkel to improve the robustness, as the Mach-number approaches 0, around stagnation points. This modification in particular limits the departure from normality of the system-matrix eigenvector system.

The aim of the present paper is to review the construction of polynomial preconditioners for conservation laws based on a convergent splitting of the Jacobi matrices. Both a simple truncation of the Neumann series and a more involved preconditioner formulated in terms of Chebyshev polynomials are considered. It is shown that the polynomial preconditioner based on Chebyshev polynomials arrived at is optimal in a certain sense if the eigenvalues of the Jacobi matrix are real. In addition, it remains effective if the eigenvalues are complex. This optimality only holds for Jacobi matrices with a corresponding eigenvector system which does not contain 'near-dependence' [17] to which we restrict ourselves here. Other approaches for non-Hermitian matrices can be found in [18]. Furthermore, it is shown that the polynomial preconditioning is applicable to any classical implicit method based on a convergent quasi-Newton iteration and may lead to a considerable reduction in the computation time. Finally,

the efficiency of the method is illustrated with simulation results for a viscous Burgers equation and inviscid flow around an airfoil, in two spatial dimensions. It is shown that sufficiently viscous flow can benefit considerably from the preconditioning while the computation time for inviscid flow calculations can be reduced by about 20%. The paper is organized as follows. In Section 2, a polynomial preconditioner in terms of Chebyshev polynomials is derived and analyzed for unsymmetric matrices. In Section 3, applications of the preconditioners to implicit methods based on a quasi-Newton iteration are discussed. Implementation-details of the preconditioners for practical computation are also presented. In Section 4, we apply polynomial preconditioning to the Burgers equation in two spatial dimensions. In Section 5, application of the approach to inviscid flow around an airfoil is presented. Concluding remarks are collected in Section 6.

2. Polynomial preconditioners

In this section we first provide a more precise definition of the basic system of equations that needs to be solved and the role of preconditioning in this context in Subsection 2.1. Subsequently, in Subsection 2.2 we describe in some detail the construction of polynomial preconditioners and establish some of their basic analytical properties.

2.1. Preconditioning quasi-Newton methods

We consider the global setting of the implicit time-integration approach to (1) and in particular identify quasi-Newton methods and the benefits of preconditioners. If we apply the method of lines, in which only the spatial discretization is treated, the following system of ordinary differential equations results:

$$\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \mathbf{f}_i(\mathbf{u}) = 0, \tag{2}$$

where a computational grid $\{\mathbf{x}_i\}$ is introduced in $\mathbb{R}^N$, $\mathbf{u}_i(t) = u(\mathbf{x}_i, t)$ and $\mathbf{f}_i(\mathbf{u})$ denotes the numerical flux. Except in one spatial dimension the index i denotes a multi-index. In an implicit method a proper backward difference method is selected subsequently for the time discretization, which yields:

$$\sum_{j=n-k}^{n+1} \alpha_j \mathbf{u}_i^j + \Delta t \mathbf{f}_i(\mathbf{u}^{n+1}) = 0, \tag{3}$$

where k is a nonnegative integer and only the numerical flux at the new time level (denoted here by the superscript, *i.e.*, $\mathbf{u}^{n+1}$) is incorporated. For convenience in the presentation we restrict to constant time-step Δt in the implicit time-stepping methods. Two typical examples are the Euler backward scheme where $k = 0$, $\alpha_{n+1} = 1$ and $\alpha_n = -1$, and the second order backward difference method 'BDF2' where $k = 1$, $\alpha_{n+1} = \frac{3}{2}$, $\alpha_n = -2$ and $\alpha_{n-1} = \frac{1}{2}$. The time step Δt is determined by temporal accuracy requirements and, in general, may depend on n. In that case the coefficients α_j become dependent on the ratios of several previous time-steps which, however, constitutes only a technical complication which we will not address in this paper. In actual simulations the general case of nonuniform Δt is used. In order to determine the solution at the new time-level we introduce a vector $\mathbf{v}$ with $\mathbf{v}_i = \mathbf{u}_i^{n+1}$ and define functions $\mathbf{F}_i$ with $\mathbf{F}_i(\mathbf{v}) = \alpha_{n+1}\mathbf{v}_i + \Delta t \mathbf{f}_i(\mathbf{v})$. Equation (3) then reads

$$\mathbf{F}_i(\mathbf{v}) = \mathbf{g}_i, \tag{4}$$

where the right-hand side is given by $\mathbf{g}_i = -\sum_{j=n-k}^{n} \alpha_j \mathbf{u}_i^j$. The new state $\mathbf{u}^{n+1}$ hence follows as a root of (4) for which Newton iteration could be used.

The classical Newton method for (4) leads to the following iteration

$$\mathbf{v}^{k+1} = \mathbf{v}^k + A^{-1}(\mathbf{g} - \mathbf{F}(\mathbf{v}^k)), \tag{5}$$

where A is the Jacobi matrix of $\mathbf{F}$ given by

$$A_{ij} = \left(\frac{\partial \mathbf{F}_i}{\partial \mathbf{v}_j}\right)_{\mathbf{v}^k}. \tag{6}$$

In general, the computation of the inverse of A or solving the linear system $A\mathbf{x} = \mathbf{b}$ is very time consuming in two or three spatial dimensions, in particular if one uses high order spatial discretization. Moreover, the evaluation of (6) can be quite time-consuming and complex, especially for nonlinear problems. To overcome these difficulties the classical approach is to use an approximation B of A in (5) which yields a quasi-Newton iteration

$$\mathbf{v}^{k+1} = \mathbf{v}^k + B^{-1}(\mathbf{g} - \mathbf{F}(\mathbf{v}^k)). \tag{7}$$

Clearly the more accurate the approximation B the faster the convergence of the iteration (7) is expected to be. However, finding an accurate approximation B in an explicit form such that the matrix-vector product $B^{-1}\mathbf{b}$ is easily computed, is a difficult problem. To cope with this obstacle in the present paper, we develop a preconditioning technique to find an approximation of the inverse B^{-1} in an implicit form for the quasi-Newton iteration (7).

Since the conservation law (1) is often nonlinear the Jacobi matrices depend on the iteration index in the Newton process and may need to be recalculated frequently in order to retain a good convergence-rate. Hence, we must construct a preconditioner quite often and, therefore, the total computation to solve (4) will be expensive, unless the construction of the preconditioner is computationally very cheap. On the other hand, the Jacobi matrix A is usually unsymmetric and we know little about the spectrum or even eigenspace structure of the matrix. Thus, an efficient preconditioner of A is required for a fast convergence, which is usually complicated and computationally costly in its construction. These two conflicting requirements seem to limit preconditioning methods. However, the discretization of time-dependent problems by the method of lines compensates these restrictions in two ways. First, backward time discretization always contributes positive components to the diagonal of A. This contribution is often reasonably large for conservation laws, in particular if a phenomenon is simulated time-accurately and the time-step is correspondingly small. Polynomial preconditioning can benefit from this property. Second, the initial guess of the solution to (4) is often quite accurate and can be found from an extrapolation of $\mathbf{u}$ obtained at previous time levels. Usually, only a few Newton iterations are needed to obtain the solution at the next time level with sufficient accuracy. Moreover, for steady state calculations a successful approach implies the addition of a pseudo-time derivative [37] and polynomial preconditioning can in a similar way be efficient in pseudo-time as well.

Based on these observations it appears that a preconditioning which can balance between the difficulties arising from the unsymmetry of the matrices and the nonlinearity on the one hand and satisfy the requirement of a cheap construction on the other hand is desired for time-accurate simulation of conservation laws. A suitable polynomial preconditioner is one of the competitive candidates in this respect.

2.2. EXPLICIT CONSTRUCTION OF OPTIMAL POLYNOMIAL PRECONDITIONING

A way to make the quasi-Newton iteration (7) more efficient is to apply good preconditioners for the Jacobi matrices. One iteration in the quasi-Newton method implies the solution of a linear system $A\mathbf{x} = \mathbf{b}$ to which we focus attention here. In this section we start from a convergent splitting of the matrix A and construct and analyze polynomial preconditioners based either on a truncation of the Neumann series for the inverse of A or on Chebyshev polynomials.

Assume that $A = M - N$ is a convergent splitting, *i.e.*, M is nonsingular and the spectral radius $\rho(M^{-1}N) < 1$ (see [19] and [20]). The inverse of A is represented by $A^{-1} = (I - M^{-1}N)^{-1}M^{-1}$. Usually, the matrix M has a suitable structure such that the matrix-vector product $M^{-1}\mathbf{b}$ is easily computed. Our aim, therefore, is to approximate $(I - M^{-1}N)^{-1}$ in terms of a polynomial in $M^{-1}N$. Since $\rho(M^{-1}N) < 1$ we formally have

$$(I - M^{-1}N)^{-1} = \sum_{i=0}^{\infty} (M^{-1}N)^i.$$

A simple polynomial approximation of $(I - M^{-1}N)^{-1}$ is a truncated form of this Neumann series. Therefore, an approximation of A^{-1} as proposed in [11] is given by

$$G_k = \left(\sum_{i=0}^{k} (M^{-1}N)^i \right) M^{-1}. \tag{8}$$

A measure for the quality of G_k as an approximation of A^{-1} is related to the eigenvalues of $G_k A$, which all should be close to 1. We implicitly assume that the system of eigenvectors is 'sufficiently' orthogonal to ignore sensitivity of the preconditioners due to matrix non-normality (see also [17]). For a square matrix C throughout the paper the notation $\lambda(C)$ denotes an arbitrary eigenvalue of C. A straightforward computation shows that $G_k A = I - (M^{-1}N)^{k+1}$, which implies that

$$|\lambda(G_k A) - 1| \leq \tilde{\rho}^{k+1}, \tag{9}$$

where $1 > \tilde{\rho} \geq \rho(M^{-1}N)$. If G_k is used in (7) fulfilling the role of B^{-1} the basic remaining computation is the multiplication of the matrix G_k with a vector $\mathbf{b}$. This is simply done by using Horner's rule

$$G_k\mathbf{b} = (I + M^{-1}N(I + M^{-1}N(\ldots M^{-1}N(I + M^{-1}N)\ldots)M^{-1}\mathbf{b}.$$

If $\rho(M^{-1}N)$ is sufficiently smaller than 1 expression (8) gives a fair approximation for A^{-1} already at small values of the truncation order k. However, if $\rho(M^{-1}N)$ is quite close to 1 the approximation (8) requires many terms in order to be accurate and it becomes beneficial to consider alternative preconditioners.

To find a more efficient polynomial preconditioner, information about the eigenvalues of A is helpful. To establish this, let $Q(\lambda) = \sum_{i=0}^{m} a_i \lambda^i$ be the characteristic polynomial of A. It is well known that $Q(A) = 0$. If A is nonsingular then $a_0 \neq 0$ and the equality $Q(A) = 0$ implies that

$$A^{-1} = -a_0^{-1} \sum_{i=0}^{m-1} a_{i+1} A^i, \tag{10}$$

i.e., the inverse of A is expressed as a polynomial in A. Formula (10) cannot be used for numerical computation because finding the characteristic polynomial of a matrix is more difficult than the computation of the matrix inverse. However, motivated by the observation in (10), and taking into account any additional information about the eigenvalues of A one could improve the approximation (8) considerably. Let $q_k(A) = \sum_{i=0}^{k} b_i A^i$ be a polynomial preconditioner of A, where the coefficients $\{b_i\}$ are real. For notational convenience we introduce the space P_k as the set of polynomials denoted by q_k of degree at most k. It is straightforward to show that $\lambda(A)q_k(\lambda(A))$ is an eigenvalue of $A\,q_k(A)$. Our aim is to construct a polynomial approximation $q_k(A)$ of A^{-1} such that the eigenvalues of $Aq_k(A)$ are all close to 1. We will specify this construction for general unsymmetric matrices in a few steps.

First, assume for the moment that the eigenvalues of A are real and bounded by $0 < \alpha \leq \lambda(A) \leq \beta$. In the following we present a polynomial $p_k(\lambda)$ in the space P_k which is optimal in the sense that the maximal distance of an arbitrary eigenvalue of the matrix $Ap_k(A)$ from unity is minimal. A similar result can be traced back to [34]. Define $x(\lambda) = -1 + 2(\lambda - \alpha)/(\beta - \alpha)$ and the polynomial

$$p_k(\lambda) = \left(1 - \frac{T_{k+1}(x(\lambda))}{T_{k+1}(x(0))}\right) \Big/ \lambda, \tag{11}$$

where $0 < \alpha < \beta$ and $T_{k+1}(\lambda)$ is the Chebyshev polynomial of degree $k+1$. Then

$$\max_{\alpha \leq \lambda \leq \beta} |\lambda p_k(\lambda) - 1| = \min_{q_k \in P_k} \max_{\alpha \leq \lambda \leq \beta} |\lambda q_k(\lambda) - 1|. \tag{12}$$

This optimality of p_k can be established in the following way. It follows from (11) since $|x(\lambda)| \leq 1$ for all $\lambda \in (\alpha, \beta)$ that

$$\max_{\alpha \leq \lambda \leq \beta} |\lambda p_k(\lambda) - 1| = \frac{1}{|T_{k+1}(x(0))|}. \tag{13}$$

It is well-known that for any polynomial R of degree $k+1$ or less and for $|x| \geq 1$ that

$$|R(x)| \leq |T_{k+1}(x)| \max_{|t| \leq 1} |R(t)|. \tag{14}$$

Clearly, $|x(0)| = (\alpha+\beta)/(\beta-\alpha) \geq 1$ and so applying inequality (14) to $R(x(\lambda)) = \lambda q(\lambda) - 1$ at $x(0)$ gives the stated result since $|R(x(0))| = 1$ and $\max_{|t| \leq 1} |R(t)| = \max_{\alpha \leq \lambda \leq \beta} |\lambda q_k(\lambda) - 1|$.

If A is a symmetric positive definite matrix it is shown in [12] that $Ap_k(A)$ is the solution which minimizes a bound on the spectral condition number of the preconditioned matrix. However, matrices arising from discretization of conservation laws are usually unsymmetric and the eigenvalues of the matrices are not always real. To apply the polynomial preconditioning to conservation laws, first we extend the approach to matrices with complex eigenvalues. Since we are interested in the matrix of the form $I - M^{-1}N$ with $\rho(M^{-1}N) < 1$, the main result is stated in the following way.

Let $\mathcal{A} = I - \tilde{T}$ with $\rho(\tilde{T}) \leq \tilde{\rho} < 1$, where $\tilde{\rho}$ is a positive constant, and

$$r_k(\lambda) = \left(1 - \frac{T_{k+1}((\lambda - 1)/\tilde{\rho})}{T_{k+1}(-1/\tilde{\rho})}\right) \Big/ \lambda, \tag{15}$$

where $T_{k+1}(\lambda)$ is the Chebyshev polynomial of degree $k+1$. Then

$$|\lambda(\mathcal{A}r_k(\mathcal{A})) - 1| \leq \tilde{\rho}^{k+1}. \tag{16}$$

The proof of this result is somewhat lengthy and technical and is collected in the appendix. Guided by this result we can now turn to our basic problem and formulate the preconditioner for general Jacobi matrices A. Splitting the Jacobi matrix $A = M - N$ with $\rho(M^{-1}N) \leq \tilde{\rho} < 1$ we apply the polynomial (15) to approximate $(I - M^{-1}N)^{-1}$. Then an approximation of A^{-1} is given by

$$H_k = r_k(I - M^{-1}N)M^{-1} \tag{17}$$

An elementary computation shows that $H_k A = r_k(I - M^{-1}N)(I - M^{-1}N)$. Therefore, it follows from (16) that

$$|\lambda(H_k A) - 1| = |\lambda(r_k(I - M^{-1}N)(I - M^{-1}N)) - 1| \leq \tilde{\rho}^{k+1}. \tag{18}$$

It follows from the above arguments that the inequality (18) is strict if the absolute values of the imaginary parts of the eigenvalues of $M^{-1}N$ are smaller than $\tilde{\rho}$. In particular, $|\lambda(H_k A) - 1|$ can be much smaller than $\tilde{\rho}^{k+1}$ if the imaginary parts of $\lambda(M^{-1}N)$ are small. For example, if the eigenvalues of $M^{-1}N$ are real one can show that

$$|\lambda(H_k A) - 1| \leq \frac{1}{T_{k+1}(1/\tilde{\rho})} = \frac{2\tilde{\rho}^{k+1}}{(1 + \sqrt{1 - \tilde{\rho}^2})^{k+1} + (1 - \sqrt{1 - \tilde{\rho}^2})^{k+1}}, \tag{19}$$

which is indeed much smaller than $\tilde{\rho}^{k+1}$.

For practical computation we next consider how to compute the matrix-vector product $H_k\mathbf{b}$ for a given vector $\mathbf{b}$. It is well-known that Chebyshev polynomials satisfy the following 3-term recurrence relation.

$$T_0(\lambda) = 1, \quad T_1(\lambda) = \lambda, \quad T_i(\lambda) = 2\lambda T_{i-1}(\lambda) - T_{i-1}(\lambda), \quad i \geq 2. \tag{20}$$

Therefore, it follows from (15) that $r_0(\lambda) = 1$. Denote $\sigma = 1/\tilde{\rho}$ then we have for $k \geq 1$

$$\begin{aligned}
r_k(\lambda) &= \frac{1}{T_{k+1}(-\sigma)\lambda}\,(T_{k+1}(-\sigma) - T_{k+1}(\sigma(\lambda - 1))) \\
&= \frac{1}{T_{k+1}(-\sigma)\lambda}\,(-2\sigma T_k(\sigma) - T_{k-1}(-\sigma) - 2\sigma(\lambda - 1)T_k(\sigma(\lambda - 1)) + T_{k-1}(\sigma(\lambda - 1))) \\
&= \frac{1}{T_{k+1}(-\sigma)\lambda}(2\sigma(\lambda - 1)(T_k(-\sigma) - T_k(\sigma(\lambda - 1))) - (T_{k-1}(-\sigma) - T_{k-1}(\sigma(\lambda - 1))) \\
&\quad -2\sigma\lambda T_k(-\sigma)) \\
&= \frac{2\sigma T_k(-\sigma)}{T_{k+1}(-\sigma)}((\lambda - 1)r_{k-1}(\lambda) - 1) - \frac{T_{k-1}(-\sigma)}{T_{k+1}(-\sigma)}r_{k-2}(\lambda),
\end{aligned}$$

which yields the following recurrence relation for $r_k(\lambda)$

$$r_k(\lambda) = \alpha_k((\lambda - 1)r_{k-1}(\lambda) - 1) - \beta_k r_{k-2}(\lambda), \quad r_0(\lambda) = 1, \quad r_{-1}(\lambda) = 0. \tag{21}$$

where $\alpha_k = T_k(-1/\tilde{\rho})/T_{k+1}(-1/\tilde{\rho})$ and $\beta_k = T_{k-1}(-1/\tilde{\rho})/T_{k+1}(-1/\tilde{\rho})$. A similar relation was found earlier in [32] and used for practical computation in [16]. Based on the recurrence relation (21) an efficient algorithm to compute $H_k\mathbf{b}$ can be formulated.

In the numerical realization of the polynomial preconditioner based on Chebyshev polynomials the estimate of the upper-bound for the spectral radius $\tilde{\rho}$ plays an important role. As is

well known a computationally affordable (rough) estimate of an upper-bound of the spectral radius can be obtained by calculation of the 1-norm of the system matrix. In general it is a rather subtle matter to find more accurate estimates of the spectral radius which are also computationally efficient. For specific conservation laws some knowledge of properties of the solution can be used to improve the estimates and hence arrive at a further improvement of the efficiency of the method. For certain applications a specific strategy for estimating the spectral radius accurately could be adopted which renders the full method more efficient. As an example, for steady state calculations it may not be required to update the estimate for ρ every iteration. Instead, an efficient method can be arrived at by providing an accurate estimate and retain this value for many subsequent iterations. These issues of problem dependent fine-tuning will not be considered in the present paper but rather we will focus on comparing the Neumann with the (rough) Chebyshev preconditioning and identify global features of the conservation law which contribute positively to the use of the preconditioning. In particular, the dissipative character of the conservation law will be shown to be of importance.

The results formulated in this section and the conditions under which they can be derived have a close relationship with the conditions for convergence of classical iterative methods for implicit time-stepping schemes, to which we turn in the next section.

3. Polynomial preconditioning and convergent splitting

In the previous section, we presented polynomial preconditioning based on convergent splittings of matrices and Chebyshev polynomials. In this section, we proceed to apply the preconditioners in a general framework as arises in the field of Computational Fluid Dynamics (CFD). It is shown that the convergence requirement of a basic iterative method implies the condition $\rho(M^{-1}N) < 1$ which is the same as is required in the application of the preconditioners. Thus, polynomial preconditioning of an already convergent method enhances the convergence rate. As an illustration, implementational aspects of polynomial preconditioners in relation to the SSOR relaxation are presented. First, however, we illustrate the methods developed in Section 2 for a specific example.

Consider the one dimensional diffusion problem

$$\begin{cases} u_t = \varepsilon u_{xx}, \quad x \in (0,1),\ 0 < t < \infty \\ \\ u(x,0) = u_0(x), \quad u(0,t) = f(t), \quad u(1,t) = g(t). \end{cases}$$

We use a central difference scheme with a uniform mesh size Δx in space. If the Euler backward scheme is applied for the time discretization, we have

$$u_i^n - u^{n-1} = \frac{\varepsilon \Delta t}{\Delta x^2}(u_{i-1}^n - 2u_i^n + u_{i+1}^n).$$

The Jacobi matrix of the Newton iteration is of the form

$$A = I + \frac{\varepsilon \Delta t}{\Delta x^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} = M - N,$$

where $M = \mathrm{diag}(A)$ in this example. It is straightforward to show that all eigenvalues of $M^{-1}N$ are real and $\rho(M^{-1}N) \le S/(S+1) < 1$ where $S = 2\varepsilon\Delta t/\Delta x^2$. Assume $S \le c$, where c is a positive constant. If $k = 1$, then (9) gives an error $|\lambda(G_k A) - 1| \le (c/(1+c))^2$, but (19) yields a much smaller error $|\lambda(H_k A) - 1| \le c^2/(2 + 4c + c^2)$. This illustrates the possible effectiveness of the preconditioning as in (17) compared to the Neumann-series as in (8).

A classical implicit time-stepping method in conservation laws based on a Newton iteration can be generally formulated as a quasi-Newton iteration (7) with $B = M$. For example, for two-dimensional problems the Jacobi matrix A is split into the form

$$A = D + L_1 + L_2 + U_1 + U_2, \tag{22}$$

where D is the (block) diagonal of A, L_1 and U_1 are (block) lower and upper triangular matrices respectively, corresponding to the spatial discretization of derivatives in the x direction, and L_2 and U_2 are (block) lower and upper triangular matrices respectively corresponding to the spatial discretization of derivatives in the y direction. Typical approximating splittings M of A used in CFD problems are

- (block) Jacobi relaxation: $M = D$, or $M = D + L_1 + U_1$, or $M = D + L_2 + U_2$,
- Gauss–Seidel relaxation: $M = D + L_1 + L_2$, or $M = D + U_1 + U_2$,
- approximate factorization of the form
  $$M = (G + L_1 + L_2)G^{-1}(G + U_1 + U_2). \tag{}$$
  This includes incomplete factorizations. In particular, if $G = D/\omega$ is chosen with a suitable real parameter ω, we have the SSOR approximation of A, which reduces to the symmetric Gauss-Seidel relaxation if $\omega = 1$.

For linear problems of the form (1) the quasi-Newton iteration (7) reduces to a basic iterative method for the linear system with the splitting $A = M - N$. It is well known that the method is convergent if and only if $\rho(M^{-1}N) < 1$. For nonlinear problems following [35] one can prove that the convergence requirement for the quasi-Newton method implies that $\rho(M^{-1}N) < 1$ for (7) with $B = M$. Therefore, the condition $\rho(M^{-1}N) \le \tilde{\rho} < 1$ is necessary to ensure convergence of a classical implicit method based on a quasi-Newton iteration for CFD problems. This condition coincides with the requirement on the spectral radius of the splitting in the construction of the polynomial preconditioners in the previous section. Hence, the polynomial preconditioning is applicable to any convergent implicit method and may give rise to a considerable improvement of the convergence rate of the method.

To implement polynomial preconditioning to typical implicit methods in CFD the main task is to compute the matrix-vector product $M^{-1}N\mathbf{b}$. This is straightforward if Jacobi or Gauss–Seidel relaxation is applied because the matrix N is explicitly given and the linear system $M\mathbf{y} = N\mathbf{b}$ is easily solved. For an incomplete factorization we have $N = M - A$, although the entries of N are not explicitly given. Hence, $M^{-1}N\mathbf{b} = (I - M^{-1}A)\mathbf{b}$, which requires the calculation of the matrix-vector product $A\mathbf{b}$ and the solution of $M\mathbf{y} = A\mathbf{b}$. In case M is the SSOR approximation of A, we apply polynomial preconditioning in the following way to reduce the number of arithmetic operations in simulations.

Represent $A = D + L + U$, where D is the (block) diagonal of A, $L = L_1 + L_2$ and $U = U_1 + U_2$. The SSOR relaxation implies a splitting of the form

$$M = (G + L)G^{-1}(G + U), \tag{23}$$

where $G = D/\omega$ with a positive parameter ω. Write

$$A = M + D - G - LG^{-1}U = (G + L)(I - P)(I + G^{-1}U),$$

where

$$P = (G + L)^{-1}(G - D + LG^{-1}U)(G + U)^{-1}G. \tag{24}$$

It is straightforward to show that $\rho(P) = \rho(M^{-1}N)$, where $N = M - A$. Therefore, we use polynomial preconditioning to approximate $(I - P)^{-1}$ and obtain an approximation of A by

$$A_k = (I + G^{-1}U)^{-1} p_k(I - P)(G + L)^{-1}, \tag{25}$$

where $p_k(I-P)$ is a preconditioner of $I-P$. The first and the last terms on the right hand side in (25) constitute no computational problem since U and L are triangular matrices and thus the linear systems $(G + L)^{-1}\mathbf{b}$ and $(I + G^{-1}U)\mathbf{c}$ can readily be solved. In the preconditioner $p_k(I - P)$ the matrix-vector product $P\mathbf{b}$ is the basic quantity which needs to be computed efficiently. We turn to this next and with some straightforward calculations we find

$$\begin{aligned} P &= (G + L)^{-1}(G - D + (G + L - G)G^{-1}(G + U - G))(G + U)^{-1}G \\ &= (G + L)^{-1}((2 - \omega)G + (G + L)G^{-1}(G + U) - (G + L) - (G + U))(G + U)^{-1}G \\ &= I + (2 - \omega)(I + G^{-1}L)^{-1}(I + G^{-1}U)^{-1} - (I + G^{-1}L)^{-1} - (I + G^{-1}U)^{-1} \\ &= \frac{1 - \omega}{2 - \omega} I + (2 - \omega)\left(\frac{I}{2 - \omega} - (I + G^{-1}L)^{-1}\right)\left(\frac{I}{2 - \omega} - (I + G^{-1}U)^{-1}\right). \end{aligned}$$

This suggests an efficient way to implement $P\mathbf{b}$. The arithmetic operations are approximately the same as those needed to solve the linear system $M\mathbf{y} = \mathbf{c}$. Compared with the direct computation via $M^{-1}N\mathbf{b} = (I - M^{-1}A)\mathbf{b}$, we save a matrix-vector product. Because matrix-vector products like $M^{-1}N\mathbf{b}$ need to be computed many times in numerical simulations of flows the improvement using this implementation of polynomial preconditioning of $I - P$ can be considerable.

4. Application to a viscous Burgers equation

To illustrate the efficiency of polynomial preconditioning for conservation laws, in this section we apply our method to a simple model problem in CFD which originates from the Burgers equation in two spatial dimensions. We compare the efficiency of the different approaches developed in the previous sections by focusing on the calculation time.

The Burgers equation which we consider reads

$$u_t + \left(\frac{1}{2}u^2\right)_x + bu_y = \nu\Delta u, \quad (x, y) \in (0, 1) \times (0, 1), \tag{26}$$

$$\begin{cases} u(0, y, t) = \frac{3}{2}, & u(1, y, t) = -\frac{1}{2} \\ u(x, 0, t) = \frac{3}{2} - 2x, & \frac{\partial}{\partial y}u(x, 1, t) = 0 \\ u(x, y, 0) = \frac{3}{2} - 2x, \end{cases}$$

where ν and b are positive constants. When $\nu = 0$ and $b = 1$ the steady-state solution to this problem is (see Figure 1)

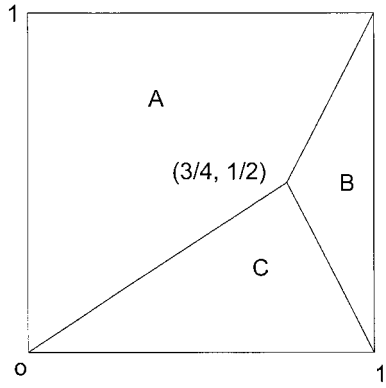

Figure 1. Representation of the steady-state solution of the Burgers equation at $\nu = 0$.

$$u(x, y) = \begin{cases} \frac{3}{2}, & \text{if } (x, y) \text{ in region } A, \\ -\frac{1}{2}, & \text{if } (x, y) \text{ in region } B, \\ \dfrac{\frac{3}{2} - 2x}{1 - 2y}, & \text{if } (x, y) \text{ in region } C. \end{cases}$$

An oblique shock starting at the point $(x, y) = (\frac{3}{4}, \frac{1}{2})$ separates regions A and B (see [36] for further details).

Using a uniform mesh-size h in both x and y directions, a uniform time step Δt, and denoting $u_{ij}^n = u(ih, jh, n\Delta t)$, we discretize the Burgers equation using the first order upwind conservative scheme for the convection term, the central difference scheme for the diffusion term, and the second order backward difference method BDF2 for the time discretization. This yields the following implicit method

$$\begin{cases} \frac{3}{2}u_{ij}^{n+1} - 2u_{ij}^n + \frac{1}{2}u_{ij}^{n-1} + \Delta t f_{ij}(\mathbf{u}^{n+1}) = 0 & 1 \le i, j \le m, \\ \frac{3}{4}u_{i,m+1}^{n+1} - u_{i,m+1}^n + \frac{1}{4}u_{i,m+1}^{n-1} + \Delta t f_{i,m+1}(\mathbf{u}^{n+1}) = 0 & 1 \le i \le m, \end{cases} \tag{27}$$

where $\mathbf{u}^{n+1} = (u_{11}^{n+1}, \ldots, u_{m1}^{n+1}, \ldots, u_{1,m+1}^{n+1}, \ldots, u_{m,m+1}^{n+1})^T$ and $m = h^{-1} - 1$. The scheme is a 3-level implicit method, which requires a separate startup procedure. For clarity we simply choose the Euler backward scheme

$$\begin{cases} u_{ij}^1 - u_{ij}^0 + \Delta t f_{ij}(\mathbf{u}^1) = 0 & 1 \le i, j \le m, \\ \frac{1}{2}u_{i,m+1}^1 - \frac{1}{2}u_{i,m+1}^0 + \Delta t f_{i,m+1}(\mathbf{u}^1) = 0 & 1 \le i \le m, \end{cases} \tag{28}$$

where the unknowns $\mathbf{u}_i^0$ are obtained from the initial data. Note that in (27) and (28) the set of equations valid as $j = m + 1$ are slight modifications of the standard BDF2 representing the Neumann condition on the boundary $y = 1$. This problem changes from a parabolic to a hyperbolic one in case ν approaches 0. In view of the use of upwind discretization and a suitable, robust treatment of the boundary condition which applies at $y = 1$, no additional special precautions are needed in the present numerical treatment. The Equations (27) and (28) are easily put in the form (4). Let A be the Jacobi matrix defined by (6). Our polynomial preconditioners are based on the Jacobi splitting $A = M - N$, where M is the diagonal of A. For this splitting it is straightforward to show that

$$\rho(M^{-1}N) \leq \|NM^{-1}\|_1 < \frac{4\nu\Delta t/h^2 + (b + 2|u_{ij}^n|)\Delta t/h}{\sigma_n + 4\nu\Delta t/h^2 + (b + |u_{ij}^n|)\Delta t/h}, \tag{29}$$

where $\sigma_0 = 1$ and $\sigma_n = 3/2$ if $n \geq 1$. Note that this bound is only a very rough estimate. In practice $\rho(M^{-1}N)$ is often much smaller than the right hand side of (29). Therefore, the condition

$$\Delta t \leq \frac{\sigma_n h}{|u_{ij}^n|} \tag{30}$$

suffices to ensure $\rho(M^{-1}N) < 1$. In view of the required time-accuracy we use a uniform time-step $\Delta t = h$. If $n \geq 1$ the property $\rho(M^{-1}N) < 1$ follows from (30) because we know $-\frac{1}{2} \leq u_{ij}^n \leq \frac{3}{2}$ from the initial and the boundary conditions of the Burgers equation and the maximum principle. For $n = 0$ it is straightforward to check $\rho(M^{-1}N) \leq \|NM^{-1}\|_1 < 1$. The stopping criterion for the quasi-Newton iteration at step $n + 1$ is determined by

$$\|F(\mathbf{v}) - \mathbf{g}\|_1 < \varepsilon(n),$$

where $\varepsilon(n)$ is conveniently chosen as $\varepsilon(n) = \min(10^{-4}, \Delta t\|\mathbf{u}_t^n\|_1/20)$. Since the equation has a steady-state solution, our numerical experiments correspond to time-accurate simulations starting from an initial solution to a steady-state solution. The stopping criterion for a steady-state solution used here is $\|u_t\|_1 < 10^{-6}$.

As a point of reference we also run this flow problem using the well-known compact-storage explicit Runge-Kutta method

$$\begin{cases} \mathbf{u}_i^{(0)} = \mathbf{u}_i^n \\ \mathbf{u}_i^{(k)} = \mathbf{u}_i^{(0)} - \Delta t_n \beta_k \mathbf{f}_i(\mathbf{u}^{(k-1)}) \quad \text{for } k = 1, 2, 3, 4 \\ \mathbf{u}_i^{n+1} = \mathbf{u}_i^{(4)} \end{cases} \tag{31}$$

with $\beta_1 = 1/4$, $\beta_2 = 1/3$, $\beta_3 = 1/2$ and $\beta_4 = 1$. For the Runge-Kutta method the time-step is determined by a stability analysis with CFL-number equal to $\sqrt{2}$. Moreover, we incorporate the standard Jacobi method in the comparison in which we solve the linear system involved in an update in the quasi-Newton process with the Jacobi scheme. Thus, we compare our preconditioning approach for an implicit method with an explicit time-stepping scheme as well as with a standard implicit scheme without preconditioning. For all implicit methods including the Jacobi method and all preconditioned methods the initial guess of the quasi-Newton process at step $n + 1$ is determined by

$$\mathbf{v}^0 = \begin{cases} \mathbf{u}^0, & \text{if } n = 0 \\ 2\mathbf{u}^n - \mathbf{u}^{n-1}, & \text{if } n \geq 1. \end{cases}$$

This extrapolation speeds up the simulation in view of the reduction in the number of quasi-Newton iteration steps.

We run all methods on a grid with $h = 1/100$ and use $b = 1$. In the simulation we vary the viscosity ν in order to assess the efficiency of the various methods and its dependence on viscosity. Because $|u_{ij}^n| \leq \frac{3}{2}$ the time step $\Delta t = h$ in all implicit methods is about $\sqrt{2}\max(5/2,\ 4\ \nu/h)$ times the time-step in the Runge-Kutta method. Table 1 shows the

Table 1. CPU-time: Runge–Kutta and Jacobi method.

ν	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	0
RK	21m26·93s	1m13·20s	0m37·36s	0m22·52s	0m22·10s	0m19·23s
J	5m46·11s	1m01·81s	0m33·90s	0m19·94s	0m18·90s	0m17·55s

Table 2. CPU-time: truncated Neumann series preconditioners.

$k \diagdown \nu$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	0
1	5m11·55s	0m57·74s	0m30·95s	0m18·00s	0m18·00s	0m17·34s
2	4m26·18s	0m54·22s	0m31·33s	0m17·70s	0m17·67s	0m16·81s
3	4m02·92s	0m50·10s	0m33·58s	0m17·90s	0m17·31s	0m17·00s
4	3m47·20s	0m56·93s	0m39·13s	0m16·57s	0m15·48s	0m15·92s
5	3m45·75s	0m51·21s	0m33·86s	0m15·34s	0m15·11s	0m15·10s
6	3m42·64s	0m53·73s	0m38·57s	0m17·37s	0m16·80s	0m16·87s
7	3m43·60s	0m55·88s	0m42·19s	0m18·85s	0m18·45s	0m18·28s
8	3m38·29s	0m55·31s	0m46·74s	0m20·58s	0m19·73s	0m19·64s
9	3m52·35s	0m58·87s	0m50·56s	0m22·31s	0m21·46s	0m22·05s
10	3m57·49s	0m53·05s	0m54·17s	0m23·26s	0m23·01s	0m22·85s

CPU-time for the Runge-Kutta method (RK) and the Jacobi method (J) needed to obtain the steady state. These calculations were performed on a standard HP 700-series workstation. Compared with RK one clearly observes a strong decrease in the CPU-time when using the Jacobi method in case the viscosity ν is relatively large. For small ν a gain of only about 5% is observed.

For the polynomial preconditioners the methods are compared with different degree of polynomials from 1 to 10. One may anticipate that an increase in the degree of the polynomial will, in general, improve the quality of the preconditioner, however, at the expense of additional calculation. By varying the degree, an idea of the optimal value may be obtained. For the preconditioned methods based on Chebyshev polynomials the bound $\tilde{\rho}$ (which is required in (15)) of the spectral radius $\rho(M^{-1}N)$ is estimated by the 1-norm of the matrix NM^{-1} at each iteration. Tables 2 and 3 show the CPU-time of the simulations using the polynomial preconditioners. It was verified separately that the computations indeed follow the solution of the Burgers equation to the steady-state accurately in time and hence these numerical experiments model actual unsteady flow simulations rather well. The relative improvement due to preconditioning is quite large in this application as can be inferred from comparing entries from Table 1 with Table 2 and Table 3. We also studied the benefit of polynomial preconditioning as a function of problem-size for this problem and observed a comparable speed-up ratio due to polynomial preconditioning for problems which are orders of magnitude larger. This suggests that application to large-scale computations, *e.g.*, involving direct and large-eddy simulation of turbulent unsteady flow in three dimensions will benefit equally from this type of preconditioning.

Table 3. CPU-time: Chebyshev polynomial preconditioners.

$k \diagdown \nu$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	0
1	3m51·08s	0m53·17s	0m39·60s	0m17·74s	0m18·27s	0m17·83s
2	2m20·97s	0m42·88s	0m37·71s	0m15·08s	0m14·99s	0m14·63 s
3	1m47·87s	0m41·15s	0m34·15s	0m16·81s	0m17·11s	0m16·07s
4	1m46·72s	0m38·98s	0m39·25s	0m17·25s	0m18·45s	0m17·39s
5	1m26·31s	0m42·30s	0m42·75s	0m20·00s	0m19·13s	0m18·71s
6	1m 21·57s	0m46·48s	0m45·51s	0m20·53s	0m20·47s	0m20·72s
7	1m35·27s	0m50·34s	0m49·47s	0m24·14s	0m22·92s	0m22·83s
8	1m39·79s	0m54·52s	0m53·34s	0m25·41s	0m25·19s	0m25·54s
9	1m42·03s	1m00·44s	0m57·03s	0m27·98s	0m27·59s	0m26·03s
10	1m47·47s	1m04·91s	1m02·35s	0m29·14s	0m29·65s	0m29·05s

As shown by the numerical results collected in Tables 1–3 polynomial preconditioners indeed can provide considerable reduction of the simulation time, in particular, if the viscosity ν is large. It appears that the preconditioners based on the Chebyshev polynomials yield better results in case the viscosity is sufficiently high. The larger the viscosity ν the bigger the improvement due to the preconditioners. If $\nu \leq 10^{-3}$ the difference in efficiency between the two polynomial preconditioners is not apparent. The reason is that in this case the spectral radius $\rho(M^{-1}N)$ is well separated from 1. Numerical experiments show that typically $\rho(M^{-1}N) < 0{\cdot}75$ if $n = 1$ and $\rho(M^{-1}N) < 0{\cdot}66$ if $n > 1$. Both (8) and (17) give fair approximations of A^{-1} already with low degree polynomials. Equation (17) usually gives a better approximation, but requires an estimate of the spectral radius of $M^{-1}N$ at each iteration and thus needs extra computation. This renders the 'Chebyshev'-method about equally efficient as the method based on the 'Neumann-series' preconditioner in case the viscosity tends to 0 . Problem dependent fine-tuning in this respect may improve the results obtained with the Chebyshev preconditioners, but this will not be considered here in order not to obscure the comparison. In some cases the extra work involved in estimating $\tilde{\rho}$ may even lead to a situation in which the Chebyshev-preconditioners yield a somewhat worse result compared to (8) as far as computation time is concerned. This is the case when $\nu = 10^{-3}$ in our numerical example. The optimal degree of the polynomials is not theoretically determined in the present paper. However, the numerical results indicate that it depends on the viscosity ν to some extent. In Tables 2 and 3 we underlined the optimal CPU-time for different viscosity ν. For the preconditioners based on Chebyshev polynomials it shows a regular pattern. The larger ν the higher the optimal degree of the polynomials. For the preconditioners based on a truncated Neumann series the situation differs considerably. The optimal degree is not very sensitive to the viscosity ν. For example, the degree $k = 3$ for the preconditioner based on Chebyshev polynomials is quite a good choice and the degree $k = 5$ for the preconditioner based on the truncated Neumann series is quite useful for $0 \leq \nu \leq 1/10$. Another clear phenomenon in Tables 2 and 3 is that the larger the viscosity the longer the CPU-time. This is caused by two main contributing factors. First, the larger the viscosity the longer it takes, measured in actual physical time measured in seconds to go from the initial solution to the steady-state solution. For example, the dimensionless real-life time in case $\nu = 1/10$ is 3·52

times that in case $\nu = 0$. Note that the real-life time is approximately the same for $\nu \leq 10^{-4}$. Second, the larger the viscosity the larger the spectral radius $\rho(M^{-1}N)$. Therefore both (8) and (17) need high degree polynomials for a fair approximation and at each time step more iterations are needed in order to obtain a sufficiently far converged solution. In this regime Chebyshev preconditioning is supposed to work better than the simple truncated Neumann series approach as is clearly illustrated by the results.

5. Application to inviscid flow around an airfoil

In this section we consider the application of polynomial preconditioning in the context of inviscid transonic flow around an airfoil. First we briefly describe the governing equations and the spatial discretization. Subsequently, we proceed with the implicit Euler backward scheme and present the basic solution method, using symmetric Gauss-Seidel relaxation. We then establish the reduction in computational effort obtained when adopting in addition a polynomial preconditioning.

The application that we study involves steady flow around an airfoil in the transonic regime. Although we are interested in the steady flow we do not solve the stationary equations directly but rather start from an initial condition and use a time stepping scheme to obtain the stationary solution. Hence, we also start from the unsteady Euler equations for inviscid compressible flow. In Cartesian coordinates in two dimensions these can be written as

$$\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0 \tag{32}$$

with

$$q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \quad f = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ u(E+p) \end{bmatrix} \quad \text{and} \quad g = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}, \tag{33}$$

where ρ is the density, E is the total energy density and u and v are the velocity components in x and y direction respectively. The constitutive equation for the pressure, p, is given by

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho(u^2 + v^2)\right), \tag{34}$$

where γ is the adiabatic gas constant which we take $\gamma = 1{\cdot}4$.

The Euler equations in (32) are in conservation form and the state vector q contains the densities of the conserved quantities and f and g are the corresponding flux vectors. Integration of (32) over an arbitrary volume in space shows that the components of q change only due to a flux contribution through the boundaries of this volume. To solve (32) we adopt a finite volume method on a structured grid which retains the conservation property. In this method we compute the flux over the control volume edges as in Figure 2 (left).

$$\Omega_{i,j}\frac{\mathrm{d}q_{i,j}}{\mathrm{d}t} + h_{i+\frac{1}{2},j} - h_{i-\frac{1}{2},j} + h_{i,j+\frac{1}{2}} - h_{i,j-\frac{1}{2}} = 0, \tag{35}$$

where $\Omega_{i,j}$ is the area of the control volume and h denotes the numerical flux vector on the four boundaries segments $(i + \frac{1}{2}, j)$ etc.

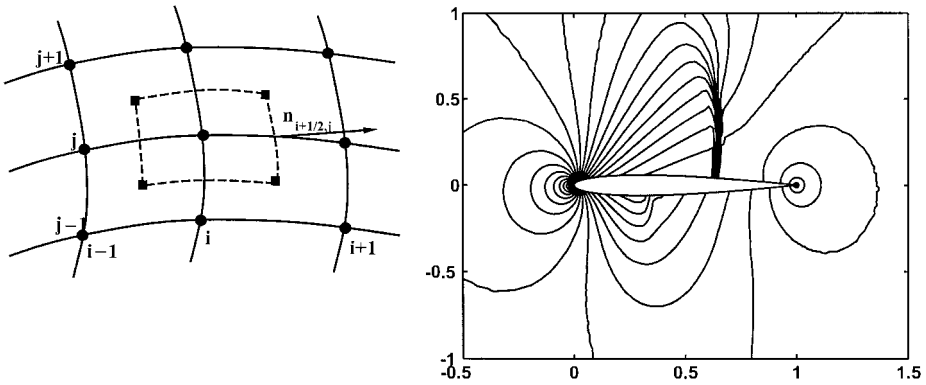

Figure 2. Left: Control volume for spatial discretization on a structured grid indicating also the outward normal vector $\mathbf{n}_{\mathbf{i}+\frac{\mathbf{1}}{\mathbf{2}},\mathbf{j}}$. Right: Contourplot of the pressure-field around a NACA0012 airfoil at transonic flow conditions.

In transonic flow applications shocks may occur which impose specific restrictions on the spatial discretization. Accurate solutions may be obtained if the shock can be properly located and if the scheme can suppress undesired numerical oscillations near the shock. By construction, scheme (35) is in conservation form which is a necessary condition for a proper capturing of the shock [3]. To suppress numerical oscillations near the shock, artificial dissipation can be added explicitly or implicitly by the use of upwind schemes. First order upwind schemes suppress these oscillations too strongly and hence smear the shock. This can be avoided if a higher order TVD (Total Variation Diminishing) scheme as *e.g.*, developed by Van Leer [31] is used. This type of higher order schemes uses a nonlinear 'limiter' function which limits differences in the gradient of the solution between adjacent grid cells. The well-known min-mod limiter is adopted in this paper [3]. The numerical flux on the control volume edges is approximated using the flux splitting method of Roe [28, 22]. The parameters in this scheme were chosen such as to ensure monotonicity [26, 24, 23] and third order accuracy in regions in which the solution is smooth.

For the flow around an airfoil there are two types of boundaries. The far field boundary arises because of the finite extent of the computational domain and solid wall conditions apply at the airfoil. In the far field subsonic inflow or outflow may occur and a method is used which takes the incoming and outgoing characteristics into account. Depending on whether the boundary is an inflow or outflow boundary we extrapolate one or three Riemann invariants from the inner field respectively and set the remaining Riemann invariants to their value at infinity [28]. The only physical condition for inviscid flow over a solid wall is the impermeability of the solid wall which is equivalent to setting the normal velocity on the solid wall equal to zero. As numerical boundary conditions we extrapolate in addition the density, the tangential velocity and the pressure from the flow field to the wall. Due to the use of a so called C-grid the trailing edge of the airfoil may become multi-valued. To prevent this we average the values of the state vectors at the trailing edge after every update of the state vector.

To initialize the flow field, we set all dependent variables equal to their values at infinity determined by the Mach number, M_∞, and the angle of attack, α. As an illustration of the method we calculated the well-known test case of inviscid flow around a NACA0012 airfoil at $M_\infty = 0{\cdot}8$ and $\alpha = 1{\cdot}25^o$ [27]. With this combination of free-stream Mach number and angle of attack the flow is transonic. The solution displays a strong shock on the upper surface of the airfoil, a weak shock on the lower surface and a weak contact discontinuity in the wake. The

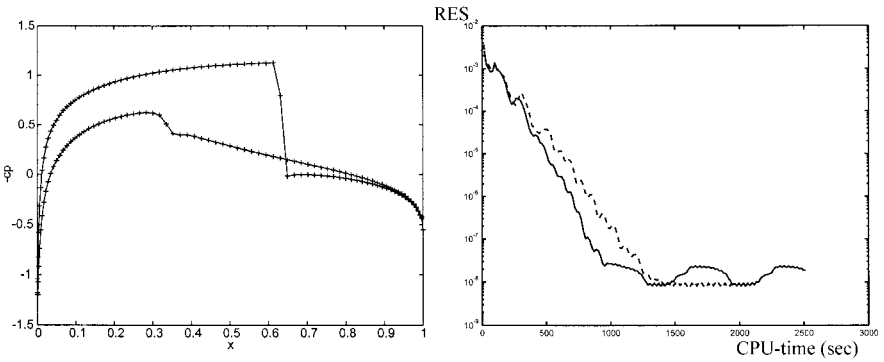

Figure 3. Left: Pressure-coefficient along a NACA0012 airfoil at transonic flow conditions. Right: Discrete L_2-norm of the residual RES of the continuity equation versus cpu-time in seconds for the implicit factorization method with (solid curve) and without (dashed curve) polynomial preconditioning.

grid we use is a C-grid with 289×65 grid points where 160 points are located on the airfoil and 65 points in the wake. In Figure 2 (right) a contour-plot of the pressure is given which is in agreement with the results in [29, 30]. The strong shock on the upper side can be captured with only one grid point in the shock. This is illustrated in Figure 3 (left) which shows the pressure-coefficient along the airfoil.

As was shown in ref [22] the computational effort needed to obtain the steady state is significantly smaller with an implicit method compared to an explicit Runge-Kutta method. With a proper preconditioning the computational method may be further accelerated. First we sketch the implicit base method and show improvements arising from simple preconditioning afterwards. The implicit base method considered here is an implicit factorization method based on an Euler backward time-discretization. The discrete version of (32) for the Euler backward scheme can be written as

$$q_{i,j}^{n+1} = q_{i,j}^{n} - \mathrm{t}F_{i,j}(q^{n+1}), \tag{36}$$

where the superscript n labels the time level and $F_{i,j}$ is the total numerical flux in the grid point (i, j). The Euler backward scheme is first order accurate in time, but this is not a concern here since we are only interested in the steady state solution and thus the overall accuracy is determined by the spatial discretization. First order Taylor expansion of F around q^n yields

$$\left(\frac{I}{\Delta t} + \frac{\partial F}{\partial q}(q^n)\right) \Delta q_{i,j} = -F_{i,j}(q^n) \tag{37}$$

where $\partial F/\partial q$ is the symbolic representation of the Jacobi matrix of F and $\Delta q_{i,j} = q_{i,j}^{n+1} - q_{i,j}^{n}$. For infinite Δt and exact Jacobi matrix this scheme is equal to Newton iteration for the system of equations $F(q) = 0$. However, it is not possible to obtain the exact Jacobi matrix at a reasonable computational cost. Therefore, we approximate the Jacobi as in [25]. Corresponding to the five point stencil of Roe's scheme in 2D we get a Jacobi matrix with five bands of 4×4-matrices. The five blocks for a grid point (i, j) are given by

$$
\begin{aligned}
D_{i,j} &= A^{+}_{i-\frac{1}{2},j} - A^{-}_{i+\frac{1}{2},j} + A^{+}_{i,j-\frac{1}{2}} - A^{-}_{i,j+\frac{1}{2}} \\
N_{i,j} &= A^{-}_{i,j+\frac{1}{2}} \\
S_{i,j} &= -A^{+}_{i,j-\frac{1}{2}} \\
E_{i,j} &= A^{-}_{i+\frac{1}{2},j} \\
W_{i,j} &= -A^{+}_{i-\frac{1}{2},j},
\end{aligned}
\tag{38}
$$

where D, N, S, E and W stand for diagonal, north, south, east and west contribution. The matrices A^+ and A^- are determined by the positive and negative eigenvalues of the flux Jacobi matrix

$$A = R\Lambda L = R(\Lambda^+ + \Lambda^-)L = R\Lambda^+ L + R\Lambda^- L = A^+ + A^-. \tag{39}$$

The 'delta formulation' in (37) allows the use of an approximation of the Jacobi matrix without changing the steady state solution. If the iteration process converges it follows from (37) that the flux equals zero in all grid points and hence the solution satisfies the stationary discrete equations.The matrix on the left hand side in (37) can be rewritten in different ways in terms of the matrices D, N, S, E and W which are the contributions to $\partial F/\partial q$ from the corresponding parts in (38). We will use the implicit factorization scheme as used in [22]:

$$\left(\frac{I}{\Delta t_{i,j}} + D + N + E\right)\left(\frac{I}{\Delta t_{i,j}} + D\right)^{-1}\left(\frac{I}{\Delta t_{i,j}} + D + S + W\right)\Delta q_{i,j} = -F(q^n_{i,j}) \tag{40}$$

which corresponds to the symmetric Gauss-Seidel relaxation. The boundary condition at the solid wall is treated explicitly in the same way as for the explicit scheme. The far field boundary condition is treated implicitly. The time step in pseudo-time for the implicit scheme is determined in the same way as the stability time step for the explicit scheme as used in Section 4. Local time stepping is used to accelerate convergence.

We can now compare the computational efficiency of the implicit base method with and without polynomial preconditioning. Since we consider inviscid flow we will only use the simpler Neumann series for the preconditioning in view of the almost equal performance of this and the Chebyshev preconditioning in case of low (or zero) viscosity as was observed in the previous section. The implementation is as described in Section 3 with $\omega = 1$. A further acceleration due to preconditioning of about 20% can be achieved for residue levels on the order of 10^{-7} as is illustrated in Figure 3 (right). In this case we used two terms in the Neumann series preconditioning (8). The increased convergence rate arising from the application of the polynomial preconditioner is mainly due to the fact that applying the preconditioner is computationally less demanding than recomputing the nonlinear right-hand side. The convergence shown in Figure 3 (right) is roughly exponential and stalls at levels on the order of 10^{-8} which is due to the finite accuracy with which real numbers are represented in the computations and the sensitivity of the predictions with respect to small perturbations. Similar stall in the convergence and even oscillations at a low level, as shown here, are typical for simulations of inviscid flow around airfoils (see [22] and references therein). We also considered incorporating more terms in the series expansion which does reduce the number of iterations further but does not lead to a significant additional reduction in the cpu-time compared to the results shown in the figure. Hence, it appears in this case that a simple polynomial preconditioning

can reduce the computational effort noticeably in large-scale inviscid flow applications which directly illustrates the findings in Section 3. In particular, since the spectral radius of the system matrix appears to be well separated from unity, already low order expansion is very effective. For viscous flow it may be expected that the preconditioning is even more effective by analogy with the findings in Section 4 and also that Chebyshev preconditioning outweighs the Neumann series approach.

6. Concluding remarks

As we have shown polynomial preconditioning can lead to improvements to implicit methods for conservation laws based on a quasi-Newton method, in particular if the governing equations have some viscous contribution to the flux. Of central importance is the fact that this preconditioning can be applied to any convergent splitting of the system-matrix A and hence preconditioning will contribute positively to any already convergent method. The specific splitting $A = M - N$ can be further exploited to enhance the effectiveness of the preconditioning. The smaller the spectral radius $\rho(M^{-1}N)$ the more efficient the polynomial preconditioners. A way to make polynomial preconditioners more efficient for conservation laws is to carefully choose some splitting approaches as described *e.g.*, in [19] and [20]. For example, the SOR splitting can be an interesting choice. Another, related option is to develop a new discretization of the conservation laws so that one can more easily obtain a good splitting for the corresponding Jacobi matrices.

The 'optimal' preconditioning, incorporating eigenvalue aspects only, arrived at in Section 2 requires an estimate of the spectral radius of the convergent splitting of A. In this paper a robust but only rough estimate was used in the simulations and it appeared that the overall efficiency of the simulation method was improved considerably compared to the case without preconditioning, even with only part of the potential of the preconditioning realized. Further improvements could be expected for some, more specific applications and the additional costs of arriving at an improved estimate of the spectral radius may well be compensated by an increase in convergence-rate for the linear system solution. If M is simple, for example when using the Jacobi or the Gauss-Seidel relaxation, only little extra computation is needed. If M is complicated, however, an accurate estimate of $\rho(M^{-1}N)$ can be quite time consuming, which may significantly influence the simulation time of the method. Some estimates on the convergence rate of basic iterative methods in [20] and [19] can be applied to overcome this difficulty. A more involved improvement would require knowledge regarding the entire eigenvector system of the matrix A. In this setting the accuracy with which the inverse of the matrix A is approximated would be known more precisely compared to the present case in which only eigenvalue information is used. However, this increase in accuracy comes at a very large price regarding computational effort and is not likely to result in a useful and efficient preconditioning method for general applications.

Time-accurate simulation of turbulent flow requires suitably small time-steps to ensure a proper capturing of the detailed elements arising in the evolution. This further enhances the efficiency of low order polynomial preconditioning and hence applies well to 3D Navier-Stokes simulations such as in direct and large-eddy simulations. The application of polynomial preconditioning for time-accurate simulation in a spatially developing three dimensional transitional and turbulent mixing layer is a subject of current research and will be published elsewhere.

Since conservation laws are usually nonlinear and the Jacobi matrix at each Newton iteration is not the same, the problem of finding an optimal degree for the polynomial preconditioner is very difficult and is best approached for every application separately. Moreover, it also depends on the splitting of the Jacobi matrix. Fortunately, the efficiency of polynomial preconditioning is not very sensitive to the degree of the polynomials. For higher degree the preconditioning is more accurate but also computationally more expensive. In total, the benefits of a higher degree were found quickly to be canceled by the additional cost of the required extra calculations. A polynomial preconditioning with lower degree, for example 1 or 2, often already provides a considerable improvement to a given implicit method. This is a main benefit of polynomial preconditioning for conservation laws.

Acknowledgements

Remarks made by one of the referees have been found to be very helpful in clarifying the benefits of polynomial preconditioning based on truncated Neumann series for nonlinear evolution problems.

Appendix

To prove the statements in (15) and (16) we proceed in two steps. The result of the first step can also be shown by using some results from literature, for instance, see [33]. We prove step one here as well for convenience to the readers. First we make the following assumption on $\mathcal{A}$: Assume that the real part of an arbitrary eigenvalue of A is positive and all eigenvalues of $\mathcal{A}$ are contained in an ellipse $\tilde{E}$ in the right half complex plane

$$\tilde{E} = \{y : y = \left(\frac{\beta+\alpha}{2} - \frac{\beta-\alpha}{2}(\cos\,\theta + \mathrm{i}\,\mu\,\sin\,\theta)/\sqrt{1-\mu^2}\right), 0 \le \theta < 2\pi\}. \tag{A1}$$

The foci of the ellipse are $(\alpha, 0)$ and $(\beta, 0)$, where $0 < \alpha < \beta$ and $0 \le \mu < 1$. With the transformation $z(y) = -1 + 2(y-\alpha)/(\beta-\alpha)$, we obtain a new ellipse

$$E = \{z :\ z = -(\cos\,\theta + \mathrm{i}\,\mu\,\sin\,\theta)/\sqrt{1-\mu^2},\ 0 \le \theta < 2\pi\}.$$

Using $\gamma = \sqrt{(1+\mu)/(1-\mu)}$ we find equivalently

$$E = \{z :\ z = -\frac{1}{2}(\gamma \mathrm{e}^{\mathrm{i}\theta} + \gamma^{-1}\mathrm{e}^{-\mathrm{i}\theta}),\ 0 \le \theta < 2\pi\}. \tag{A2}$$

Consider the polynomial

$$p_k(y) = \left(1 - \frac{T_{k+1}(z(y))}{T_{k+1}(z(0))}\right) \Big/ y. \tag{A3}$$

Since the Chebyshev polynomial $T_{k+1}(z)$ can be written as

$$\begin{aligned} T_{k+1}(z) &= \frac{1}{2}((z+\sqrt{z^2-1})^{k+1} + (z+\sqrt{z^2-1})^{-(k+1)}) \\ &= -\frac{1}{2}(\gamma^{k+1}\mathrm{e}^{(k+1)\mathrm{i}\theta} + \gamma^{-(k+1)}\mathrm{e}^{-\mathrm{i}(k+1)\theta}), \end{aligned}$$

the maximum absolute value of $|T_{k+1}(z)|$ on the ellipse E is given by

$$\max_{z\in E} |T_{k+1}(z)| = \frac{1}{2}(\gamma^{(k+1)} + \gamma^{-(k+1)}), \tag{A4}$$

which is taken for $\theta = 0$. Hence

$$\max_{y\in\tilde{E}} |yp_k(y) - 1| = \frac{T_{k+1}(1/\sqrt{1-\mu^2})}{T_{k+1}((\alpha+\beta)/(\beta-\alpha))} \tag{A5}$$

In the second step we let $\lambda = a + i\,b$ be an eigenvalue of $\tilde{T}$, where a and b are real with $\sqrt{a^2+b^2} \le \tilde{\rho}$. Therefore, all eigenvalues of $\mathcal{A}$ are contained in the ellipse

$$P = \{x :\ x = 1 - \tilde{\rho}(\cos\,\theta + \mathrm{i}\,\sin\,\theta),\ 0 \le \theta < 2\pi\}.$$

Let ε be a small positive real number such that $\tilde{\rho} + \varepsilon < 1$. Then the ellipse P is contained in the ellipse

$$J(\varepsilon) = \{x :\ x = 1 - (\tilde{\rho}+\varepsilon)(\cos\theta + \mathrm{i}\,\mu(\varepsilon)\,\sin\theta),\ 0 \le \theta < 2\pi\},$$

where $\mu(\varepsilon) = \tilde{\rho}/(\tilde{\rho}+\varepsilon)$. It is straightforward to show that the linear system

$$\alpha + \beta\ = 2,$$

$$\beta - \alpha\ = 2(\tilde{\rho}+\varepsilon)\sqrt{1-\mu(\varepsilon)^2}$$

has a unique solution with positive $\alpha(\varepsilon)$ and $\beta(\varepsilon)$. Hence, the ellipse $J(\varepsilon)$ is of the form $\tilde{E}$. It follows from (A5) that

$$\max_{y\in J(\varepsilon)} |yp_k(y) - 1| \le \frac{T_{k+1}(1/\sqrt{1-\mu(\varepsilon)^2})}{T_{k+1}(1/((\rho+\varepsilon)\sqrt{1-\mu(\varepsilon)^2}))}. \tag{A6}$$

Setting $\varepsilon \to 0$ we find $\max_{y\in J(\varepsilon)} |yp_k(y) - 1| = \max_{y\in P} |yr_k(y) - 1|$. Therefore,

$$\max_{y\in P} |yr_k(y) - 1| \le \lim_{\varepsilon\to 0} \frac{T_{k+1}(1/\sqrt{1-\mu(\varepsilon)^2})}{T_{k+1}(1/((\tilde{\rho}+\varepsilon)\sqrt{1-\mu(\varepsilon)^2})} = \tilde{\rho}^{k+1},$$

which implies the desired result as stated above.

References

1. M. M. Rai, P. Moin. Direct numerical simulation of transition and turbulence in a spatially evolving boundary layer. *J. Comp. Phys.* 109 (1993) 169–192.
2. A. Jameson, Time dependent calculations using mulitgrid, with applications to unsteady flows past airfoils and wings. *AIAA paper: 91–1596* (1991).
3. R. J. LeVeque, *Numerical Methods for Conservation Laws.* Basel: Birkhäuser Verlag (1992) 214 pp.
4. A. Jameson, W. Schmidt and E. Turkel, Numerical solutions of the Euler equations by finite volume methods with Rundge–Kutta time stepping schemes. *AIAA–paper: 81-1259* (1981).
5. A. W. Vreman, B. J. Geurts and J. G. M. Kuerten, Comparison of subgrid–models in large eddy simulation of the temporal mixing layer. *J. Fluid Mech.* 339 (1997) 357–390.
6. B. Geurts, B. Vreman and H. Kuerten, Comparison of DNS and LES of transitional and turbulent compressible flow: flat plate and mixing layer. *Application of Direct and Large Eddy Simulation to Transition and Turbulence*, In: AGARD Conference Proceedings 551 (1994) 1–14.
7. B. Wasistho, B. J. Geurts, J. G. M. Kuerten, Accurate boundary conditions for time-dependent flow over a flat plate in 2D. *Comp. Fluids* 26 (1997) 713–735.
8. E. Turkel, Preconditioned methods for solving the incompressible and low speed compressible equations. *J. Comp. Phys.*, 72, (1987) 277–309.
9. R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. 2nd Edition, Philadelphia: *SIAM* (1994) 57–78.

10. Y. Saad, Krylov supspace methods on supercomputers. *SIAM J. Sci. Stat. Comp.* 10 (1989) 1200–1214.
11. P. F. Dubois, A. Greenbaum and G. H. Rodrigue, Approximating the inverse of a matrix for use in iterative algorithms on vector processors. *Computing* 22 (1979) 357—268.
12. O. G. Johnson, C. A. Micchelli and G. Paul, Polynomial preconditioner for conjugate gradient calculations. *SIAM J. Numer. Anal.* 20 (1983) 362–376.
13. P. Concus and G. H. Golub and G. Meurant, Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comp.* 6 (1985) 220–252.
14. H. Lu, *Compensative Block Incomplete Factorizations*. Report 9610, Department of Mathematics, University of Nijmegen, The Netherlands (1996) 1–17.
15. O. Axelsson and P. S. Vassilevski, Algebraic multilevel preconditioning methods, II. *SIAM J. Num. Anal.* 27 (1990) 1569–1590.
16. Y. Saad, Practical use of polynomial preconditionings for the conjugate gradient method. *SIAM J. Sci. Stat. Comp.* 6 (1985) 865–881.
17. D. L. Darmofal and P. J. Schmid, The importance of eigenvectors for local preconditioners of the Euler equations. *J. Comp. Phys.* 127 (1996) 346–381.
18. M. Eiermann, W. Niethammer, R. S. Varga, Acceleration of relaxation methods for non-Hermitian linear systems. *SIAM J. Matrix Anal. Appl.* 13 (1992) 979–991.
19. R. Varga, *Matrix Iterative Analysis*. Englewood Cliffs NJ: Prentice Hall (1962) pp. 47–48.
20. D. Young, *Iterative Solution of Large Linear Systems*. New York: Academic Press (1971) pp. 174–185.
21. S. Yoon and A. Jameson, Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes Equations. *AIAA J.* 26 (1988) 26–45.
22. R. van Buuren, J. G. M. Kuerten and B. J. Geurts, Instabilities of stationary inviscid compressible flow around an airfoil. *J. Comp. Phys.* 138 (1997) 520–539.
23. A. Harten and S. Osher, Uniformly high-order accurate non-oscillatory schemes I. *SIAM J. Num. Anal* 24 (1987) 279–309.
24. P. K. Sweby, High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM J. Num. Anal.* 21 (1984) 995–1011.
25. H. C. Yee, Construction of explicit and implicit shock capturing methods. *J. Comp. Phys.* 68 (1987) 151–179.
26. H. C. Yee, *Computational Fluid Dynamics*. Von Karman Institute for Fluid Dynamics, Lecture Series 1989–04 (1989) 42 p.
27. H. Yoshihara and P. Sacher, Test Cases for Inviscid Flow Field Methods. AGARD–AR–211 (1985) 34 p.
28. P. L. Roe, Characteristic-based Schemes for the Euler equations. *Ann. Rev. Fluid Mech.* 18 (1986) 337–365.
29. J. W. van der Burg, *Numerical Methods for Transonic Flow Calculations*. Phd. thesis, University of Twente, The Netherlands (1992) 224 p.
30. V. Venkatakrishnan, Convergence to Steady State Solutions of the Euler Equations on Unstructured grids with limiters. *J.Comp.Phys.* 118 (1995) 120–130.
31. B. van Leer, Towards the ultimate conservative difference scheme II. Monotonicity and conservation combined in a second-order scheme. *J. Comp. Phys.* 14 (1974) 361–370.
32. E. L. Stiefel, Kernel polynomials in linear algebra and their applications. U.S. Nat. Bur. Standards. *Appl. Math. Series* 49 (1990) 297–315.
33. B. Fischer, R. Freund, On the constrained Chebyshev approximation problem on ellipses. *J. Appr. Theory* 62 (1990) 397–415.
34. W. Markoff, Über Polynome, die in einem gegebenen Intervalle möglichst wenig von Null abweichen. *Math. Ann.* 77 (1916) 213–258.
35. R. S. Dembo, S. C. Eisenstat and T. Steihaug, Inexact Newton methods. *SIAM J. Num. Anal.* 19 (1982) 400–408.
36. S. Spekreijse, Multigrid solution of monotone second-order dicretizations of hyperbolic conservation laws. *Math. Comp.* 49 (1987) 135–155.
37. A. Jameson, Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. *AIAA paper 91-1596* (1991) 47 p.
38. P. Wesseling, High performance computing in fluid dynamics. In: P. Wesseling (ed.) Proceedings of the Summerschool on High Performance Computing in Fluid Dynamics held at Delft University of Technology. Dordrecht. Kluwer Academic Publishers (1996) 278 p.