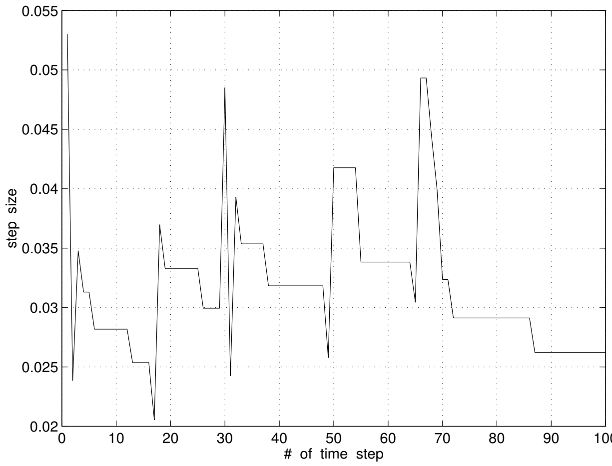

Universiteit Utrecht

Department of Mathematics

Stability Control for Approximate Implicit Time-stepping Schemes with Minimal Residual Iterations

by

Mikhail A. Botchev, Gerard L.G. Sleijpen, and Henk A. van der Vorst

Preprint

nr. 1043

December, 1997

Also available at URL `http://www.math.ruu.nl/publications/`

Stability Control for Approximate Implicit Time-stepping Schemes with Minimal Residual Iterations

Mikhail A. Botchev* Gerard L. G. Sleijpen*
Henk A. van der Vorst*

December 4, 1997

Abstract

Implicit schemes for the integration of ODE's are popular when stability is more of concern than accuracy, for instance for the computation of a steady state solution. However, in particular for very large systems the solution of the involved linear systems may be very expensive. In this paper we study the solution of these linear systems by a moderate number of iterations of the minimum residual iterative method GMRES. Of course, this puts limits to the step size since these approximate schemes may be viewed as explicit schemes and these are never unconditionally stable. It turns out that even a modest degree of approximation allows rather large time steps and we propose a simple mechanism for the control of the step size with respect to stability.

Keywords: Time-stepping schemes; Stability step size control; GMRES

1 Introduction

Implicit schemes for the integration of ODE's are popular when stability is more of concern than accuracy, for instance for the computation of a steady state solution. However, in particular for very large systems the solution of the involved linearized systems may be very expensive. In this paper we study the solution of these linear systems by a moderate number of iterations of the minimum residual iterative method GMRES [13, 1]. Of course, this puts limits to the step size since these approximate schemes may be viewed as explicit schemes and these are never unconditionally stable. It turns out that even a modest degree of approximation allows rather large time steps

*P.O.Box 80.010, 3508 TA Utrecht, the Netherlands. E-mail: [botchev, sleijpen, vorst]@math.ruu.nl

and we propose a simple mechanism for the control of the step size with respect to stability.

The usage of a few steps of GMRES is attractive, since this involves only matrix-vector operations. Specially on parallel computers this may be an advantage because matrix vector operations are usually easy to parallelize. If one uses k steps of GMRES, then the resulting approximate method may be viewed as an explicit integration scheme and such schemes have been studied heavily. Most often, these schemes are studied with fixed coefficients, while GMRES leads to different coefficients for each time step. The usage of GMRES for the approximate solution of the involved linear systems is not new either, but the resulting schemes have been studied from an accuracy point of view, that is the main focus is on strategies to obtain bounded residuals [5, 4, 3]. Another approach, that is based on Krylov subspace information, is to use an approximation for the vector $e^{\Delta t A}\boldsymbol{y}^n$ (where A is the Jacobian), for accurate time integration of stiff systems [7, 8].

In more conventional approaches for approximating an implicit scheme with explicit schemes one works with Chebyshev approximations [11, 21, 10]. Performance of these methods depends on a priori knowledge of a region containing the spectrum of the Jacobian.

We study the usage of a few steps of GMRES from the point of view of stability. A nice aspect of GMRES is that it constructs implicitly an integration polynomial of which the coefficients are adjusted to the specific right-hand side. If after some time steps components in the solution in higher frequencies have not sufficiently damped out, then they are present in the right-hand side. As soon as these components are too large then they are automatically damped out by the GMRES polynomial, provided that the time step is not too large with respect to the number of GMRES steps. We propose an easy strategy to control the size of the time step. This strategy is based on information that we obtain from the GMRES process itself. The stability control is to be applied in time stepping when the local error is not controlled, as happens typically in large physical simulation codes, e.g. [15]. However, our approximated implicit schemes can be used in an ODE code as well. Of course, there may be phases in the integration process where the local errors have too be small and where the step size is restricted by tolerances set on the local error, rather than by stability. Although our schemes have not been designed to cope with this situation specially—there is a huge variety of specially tuned schemes for this problem—it is of interest to know how our scheme competes with the existing schemes for general systems of ODE's, e.g. [3, 14]. As we will see from our numerical experiments, our scheme is more or less competitive with the existing codes when the local error is the constraining parameter, but our scheme is certainly competitive when stability is the only restriction on the time step.

The remainder of our paper has been organized as follows. Our basic approach has been formulated in Section 2. In the third section we present the stability analysis and discuss how the stability can be monitored. We will do this for two different schemes. Implementation issues are discussed in Section 4. Numerical experiments are presented in the fifth section, and we have listed some conclusions at the end.

2 Minimal Residual Approximated Implicit schemes

The general idea behind our approach is as follows. Our aim is to obtain a cheap alternative for an implicit integration scheme, without loosing too much in terms of stability. We start with a given implicit scheme of order p in time, and we approximate the solution of the associated implicit system with a few steps of a minimum residual iterative solver, say GMRES. This leads to a new scheme, and in order to let this new scheme have at least the same order of consistency, we need to start GMRES with the result of an explicit scheme of order p. This is different from the popular predictor–corrector approach, where the corrector is solved by successive substitution. This is essentially a Richardson type of iteration which leads to an integration scheme with limited stability properties comparable to the stability of the predictor.
We will work out this idea in more detail for the simple Euler Backward scheme for the system of ODE's

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(t, \boldsymbol{y}), \qquad \boldsymbol{y}\big|_{t=0} = \boldsymbol{y}^0 \in \mathbb{R}^N. \tag{1}$$

Euler Backward is given by:

$$\boldsymbol{y}_B^{n+1} = \boldsymbol{y}_n + \Delta t \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_B^{n+1}). \tag{2}$$

In order to solve this equation for $\boldsymbol{y}_B^{n+1}$, we take Euler Forward

$$\boldsymbol{y}_F^{n+1} = \boldsymbol{y}^n + \Delta t \boldsymbol{f}(t_n, \boldsymbol{y}^n), \tag{3}$$

as an initial guess, and we write

$$\boldsymbol{y}_B^{n+1} = \boldsymbol{y}_F^{n+1} + \Delta \boldsymbol{y}. \tag{4}$$

The vector $\Delta\boldsymbol{y}$ satisfies

$$\boldsymbol{y}_F^{n+1} + \Delta\boldsymbol{y} = \boldsymbol{y}^n + \Delta t \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_F^{n+1} + \Delta\boldsymbol{y}). \tag{5}$$

This leads to

$$\begin{aligned} \Delta\boldsymbol{y} &= \boldsymbol{y}^n + \Delta t \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_F^{n+1} + \Delta\boldsymbol{y}) - \boldsymbol{y}_F^{n+1} \\ &\approx \boldsymbol{y}^n + \Delta t \left\{ \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_F^{n+1}) + A\Delta\boldsymbol{y} \right\} - \boldsymbol{y}_F^{n+1}, \end{aligned}$$

where A denotes the Jacobian, evaluated in $(t_{n+1}, \boldsymbol{y}_F^{n+1})$. In the predictor–corrector approach one ignores the $\Delta \boldsymbol{y}$ in the right-hand side, and one repeats the preceding steps, which leads to an iterative application of (4):

$$\boldsymbol{y}_{(j)}^{n+1} = \boldsymbol{y}_{(j-1)}^{n+1} + \Delta \boldsymbol{y}_{(j-1)}. \tag{6}$$

In our approach, we take for $\Delta \boldsymbol{y}$ the approximate solution from

$$(I - \Delta t A)\boldsymbol{x} = \boldsymbol{r}^n \equiv \boldsymbol{y}^n - \boldsymbol{y}_F^{n+1} + \Delta t \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_F^{n+1}). \tag{7}$$

Note that $\boldsymbol{r}^n$ can be interpreted as the residual for (2), that we get when $\boldsymbol{y}_F^{n+1}$ is inserted.
We solve (7) by k steps of GMRES, with starting solution $\boldsymbol{x}_0 = 0$, so that the approximated solution $\boldsymbol{x}_k$ can be represented as

$$(\Delta \boldsymbol{y} :=) \quad \boldsymbol{x}_k = P_{k-1}^{(n)}(I - \Delta t A)\boldsymbol{r}^n,$$

where $P_{k-1}^{(n)}$ is the so-called iteration polynomial of degree $k-1$, associated with GMRES (in the case that A is symmetric, we take the MINRES method instead of GMRES). The number of iterations k is kept fixed. The superindex $\cdot^{(n)}$ has been included in order to indicate that we usually get a different polynomial for each time step. We will refer to the resulting scheme as an *Minimum Residual Approximated Implicit (MRAI)* scheme. Obviously, we can interpret the combination of (7) with GMRES as an inexact Newton method for (2).
We leave it to the reader to verify that the order of the local error is the same as for Euler Backward or Euler Forward, irrespective the number of GMRES steps used for the approximation of $\boldsymbol{x}$. For more general implicit schemes, one has to take care that the starting vector is constructed with an explicit scheme of at least the same order in time.

3 Stability of the MRAI scheme

The MRAI scheme of the previous section can be represented by

$$\boldsymbol{y}_M^{n+1} = \boldsymbol{y}_F^{n+1} + P_{k-1}^{(n)}(I - \Delta t A)(\boldsymbol{y}_M^n - \boldsymbol{y}_F^{n+1} + \Delta t \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_F^{n+1})) \tag{8}$$

with $\boldsymbol{y}_F^{n+1} = \boldsymbol{y}_M^n + \Delta t \boldsymbol{f}(t_n, \boldsymbol{y}_M^n)$.
This explicit scheme is difficult to analyze, because the polynomial $P_{k-1}^{(n)}$ depends on spectral properties of the matrix as well as on the vector on which it is acting. This means that we can not simply apply the MRAI scheme to a linear homogeneous system and consider its effect in each eigenvector direction independently.
For the stability we look at the recursions obtained when $\boldsymbol{f}(t, \boldsymbol{y}) = A\boldsymbol{y}$, that

is we consider, as is usual, the behavior for the linear homogeneous case. Form straight forward evaluation of (7), together with (3), we obtain

$$\boldsymbol{y}_M^{n+1} = (I - \Delta t A)^{-1} \left(I - \left[I - (I - \Delta t A) P_{k-1}^{(n)} (I - \Delta t A) \right] (\Delta t A)^2 \right) \boldsymbol{y}_M^n. \tag{9}$$

The polynomial $I - (I - \Delta t A)P_{k-1}^{(n)}(I - \Delta t A)$ is just the k-degree GMRES polynomial $R_k^{(n)}(I - \Delta t A)$, that is the polynomial that describes how the initial residual $\boldsymbol{r}^n \equiv \boldsymbol{r}_0^n$ is reduced to $\boldsymbol{r}_k^n$, after k steps of GMRES.
Assuming that $A \in \mathbb{R}^{N \times N}$ can be diagonalized as $A = S \Lambda S^{-1}$, where Λ is a diagonal matrix with real eigenvalues λ_j on its main diagonal, we arrive, with $\boldsymbol{z}^n \equiv S^{-1} \boldsymbol{y}_M^n$, at

$$\boldsymbol{z}^{n+1} = (I - \Delta t \Lambda)^{-1} \left(I - R_k^{(n)} (I - \Delta t \Lambda)(\Delta t \Lambda)^2 \right) \boldsymbol{z}^n, \tag{10}$$

or for the j-th component $z^n = z_j^n$ of $\boldsymbol{z}^n$, and the j-th eigenvalue $\mu = \Delta t \lambda_j$:

$$z^{n+1} = (1 - \mu)^{-1} \left(1 - R_k^{(n)} (1 - \mu) \mu^2 \right) z^n. \tag{11}$$

We have written the above expressions for the GMRES polynomial $R_k^{(n)}$, rather than for $P_{k-1}^{(n)}$, since $R_k^{(n)}$ satisfies an optimality property: $\|R_k^{(n)}(I - \Delta t A)\boldsymbol{r}^n\|_2$ is minimal over all polynomials R of degree k, satisfying $R(0) = 1$. Note that *exactly* the same reccurences as (10) and (11) describe how the initial residual $S^{-1}\boldsymbol{r}^{n+1}$ is related to its precessor $S^{-1}\boldsymbol{r}^n$.
From experiences with GMRES, it is well-known that this polynomial attempts to reduce the largest components of $S^{-1}\boldsymbol{r}^n$, by putting a root close to the corresponding eigenvalue. Therefore, if the scheme leads to a large component in the vector $S^{-1}\boldsymbol{r}^n$, because of instability, then this component will be damped and in the next time step a new polynomial is automatically constructed that tries to reduce other new dominating components. The heuristic argument behind our stability control strategy is that we assume that in average, over a number of successive time steps, no component of the vector $S^{-1}\boldsymbol{r}^n$ dominates, or in other words, that all its components are more or less equal in absolute value.

The j-th component of $S^{-1}\boldsymbol{r}^n$ is multiplied in step n by a factor to $R_k^{(n)}(1 - \mu)\mu^2$, and following the above heuristics, it is likely that the polynomial

$$Q(\mu) \equiv \mu^2 R_k^{(n)}(1 - \mu)$$

has almost the minimax property over the interval $[\Delta t \lambda_N, \Delta t \lambda_1]$ (we assume that the eigenvalues have been ordered as $\lambda_N \leqslant \ldots \leqslant \lambda_1$). Note that this leads to a weaker condition than the requirement that the reduction polynomial is small over the entire spectrum in *each* time step. The latter requirement is made for schemes based upon Chebyshev polynomial approximations.

We will further assume that $\lambda_1 < 0$. The stability condition now reads

$$\left|\frac{1 - R_k^{(n)}(1-\mu)\mu^2}{1-\mu}\right| \leqslant 1, \qquad \text{or}$$
$$\left|\frac{1 - Q(\mu)}{1-\mu}\right| \leqslant 1. \tag{12}$$

The last condition in (12) is equivalent to

$$\mu \leqslant Q(\mu) \leqslant 2 - \mu. \tag{13}$$

Note that $Q(1) = R_k^{(n)}(0) = 1$, and 0 is a root of Q of multiplicity 2. The other k roots η_i satisfy $R_k^{(n)}(1-\eta_i) = 0$. Suppose that $\eta_k \leqslant \ldots \leqslant \eta_1$. Since $\mu < 0$ over the interval of interest, we have that η_i, in particular $\eta_1 < 0$.
In Figure 1 we have plotted the situation when Q has the exact minimax property. With the estimate

$$|Q(\mu)| = \mu^2 \left|\prod_{i=1}^{k} \frac{\mu - \eta_i}{1-\eta_i}\right| \leqslant \mu^2 \frac{\mu - \eta_1}{1-\eta_1}, \qquad \mu \in [\eta_1\,;\,0), \tag{14}$$

it is possible to show (see [2] for a detailed proof) that (13) holds when

$$-7 \leqslant \eta_1 \qquad (< 0). \tag{15}$$

In fact, condition (15) guarantees that the rightmost local maximum of $Q(\mu)$ (cf. Figure 1) is below the line $2 - \mu$. The other local extrema should be between μ and $2 - \mu$. In other words, condition (15) works for polynomials $Q(\mu)$ which may only very roughly resemble the minimax polynomial. To see whether our conclusions are confirmed in practice, in Figure 3 we have plotted $S^{-1}\boldsymbol{r}^n = \boldsymbol{r}^n$ and $Q(\mu)$ observed during numerical integration of the model problem

$$\begin{gathered} \boldsymbol{y}' = A\boldsymbol{y}, \quad \boldsymbol{y}^0 = (1,\ldots,1)^T \in \mathbb{R}^{500}, \\ A = \mathrm{Diag}(-1 : 0.002 : -0.01). \end{gathered} \tag{16}$$

Here we used the descibed MRAI scheme based on Euler Backward with $k = 5$ steps of GMRES and the step size chosen to have $-7 \leqslant \eta_1 \leqslant -6.5$. It turns out that the condition (13) is true for most time steps; each time step where for some μ's the condition (13) is not fulfilled is followed by one or two “safe” steps. As we see, our assumptions leaded to realistic stability control.

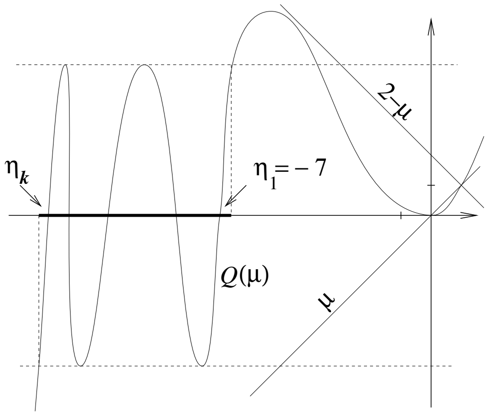

Figure 1: Stability governed by the roots of $Q(\mu)$ (cf. (13),(15))

The roots η_i of $R_k^{(n)}$ are easily obtained from the GMRES process: they are the so-called harmonic Ritz values [12]. The right-most root η_1 indicates how important the spectral information is for eigenvalues closest to the origin, and we simply have to check that this stays below -7. If $\eta_1 < -7$, then we have to decrease the step size (see Section 4).
Condition (15) is sharp for the case $k = 1$, and remains to be close to sharp for small $k > 1$. In Figure 2 we have plotted $\|\boldsymbol{y}\|_\infty$ of the computed numerical solution of the model problem (16). A decrease of the norm in the plots indicates that the scheme is stable.
For the upper plot, the solution was obtained with only $k = 1$ GMRES iteration and, at each time step, the step size Δt was chosen such that $-7.0 \leqslant \eta_1 \leqslant -6.8$). For the lower plot, we have repeated the computation with the requirement that $-7.2 \leqslant \eta_1 \leqslant -7.05$. We see that even a modest change in the bound for η_1 reflects the instability.
Other numerical experiments suggest that stability conditions similar to (15) can be applied for the case where the spectrum is in the left complex half-plane.

Another simple bound for the time step can be derived under the assumption that $R_k^{(n)}(1-\mu)$ resembles in average the k-th degree Chebyshev polynomial T_k shifted to a segment containing all μ's, for instance $[-\Delta t\lambda_N\,;\,0]$. This polynomial T_k is also scaled to be 1 at 0 (recall that $R_k^{(n)}(0) = 1$. The largest root of R_k is $1 - \eta_k$. Since we know η_k from the GMRES process, we can approximate μ_N using the relation

$$\Delta t\lambda_N = \mu_N \approx \frac{2\eta_k}{1 + \cos(0.5\pi/k)}.$$

Of course, this estimate may not be accurate. Nevertheless, it gives a good impression of the largest *effective* eigenvalue that restricts the step size.
Moreover, the stability condition (12) is satisfied when $|R_k(1-\mu)|\,|\mu| \leqslant 1$.

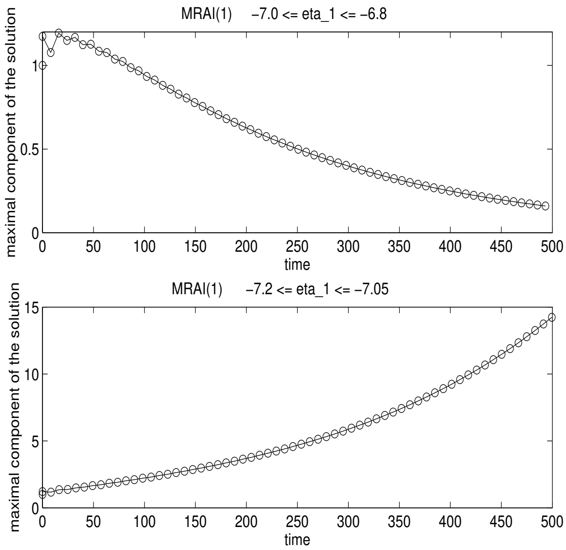

Figure 2: The maximal component of the numerical solutions. (Growth means unstability)

Then, from the estimate $|R_k(1-\mu)| \leqslant 2\left(\frac{\sqrt{1-\mu_N}-1}{\sqrt{1-\mu_N}+1}\right)^k$, we obtain

$$2\left(\frac{\sqrt{1-\mu_N}-1}{\sqrt{1-\mu_N}+1}\right)^k |\mu_N| \leqslant 1. \tag{17}$$

The last inequality can be solved numerically, which gives an upper bound $\mathcal{C}$ for $|\Delta t \lambda_N|$, and, hence, also for Δt:

$$\Delta t|\lambda_N| = |\mu_N| \leqslant \mathcal{C} \quad \text{so that} \quad \Delta t \leqslant \mathcal{C}/|\lambda_N|. \tag{18}$$

It should be remarked that the estimate (17) is typically stronger than (12), and in practice it should be relaxed, especially for small k ($k \lesssim 3$). For instance, the constant 2 is often too crude in (17), and does not reflect realistically observed values.

3.1 An example of a higher order MRAI scheme

Of course, the ideas outlined in the previous section may also be applied to other implicit schemes. In this section we discuss briefly the situation for the trapezoidal rule

$$\boldsymbol{y}_T^{n+1} = \boldsymbol{y}^n + \Delta t \boldsymbol{f}(t_{n+1/2}, \frac{1}{2}(\boldsymbol{y}^n + \boldsymbol{y}_T^{n+1})), \tag{19}$$

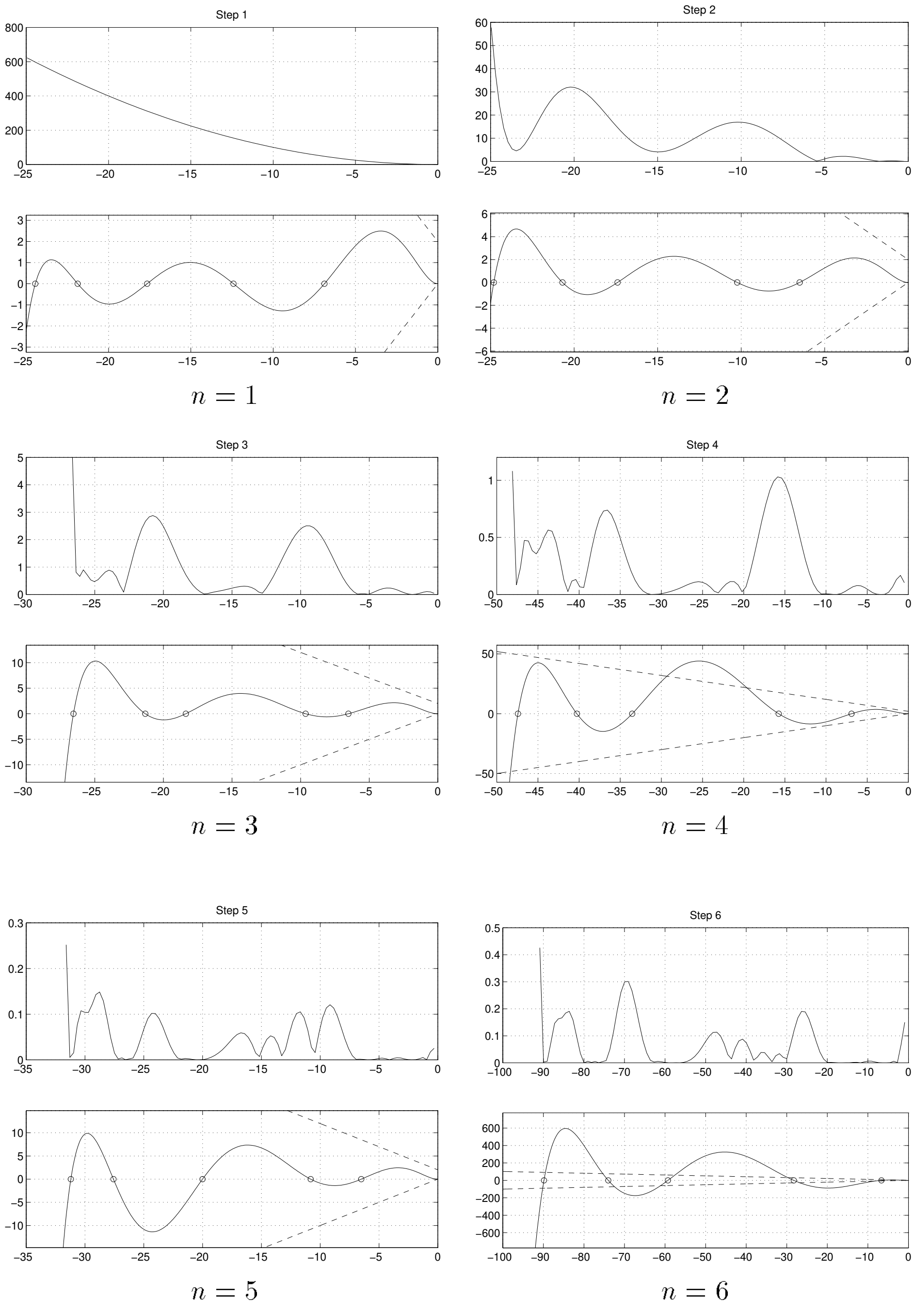

Figure 3: The absolute values of components of $S^{-1}\boldsymbol{r}^n$ plotted against the μ values and polynomials $Q(\mu)$, observed within first 6 successive time steps. The lines μ and $2-\mu$ are dashed.

where $t_{n+1/2} = t_n + \frac{\Delta t}{2}$. This scheme is the second order accurate. We consider as before the linearized form:

$$(I - \frac{\Delta t}{2}A)\boldsymbol{y}_T^{n+1} = (I + \frac{\Delta t}{2}A)\boldsymbol{y}^n + \Delta t \boldsymbol{g}^{n+1/2}, \tag{20}$$

with

$$A = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\right](t_{n+1/2}, \boldsymbol{y}^n), \quad \boldsymbol{g}^{n+1/2} = \boldsymbol{f}(t_{n+1/2}, \boldsymbol{y}^n) - A\boldsymbol{y}^n.$$

For the starting vector for the GMRES scheme, we select the second order explicit scheme

$$\boldsymbol{y}_P^{n+1} = \boldsymbol{y}^n + (I + \frac{\Delta t}{2}A)\Delta t \boldsymbol{f}(t_{n+1/2}, \boldsymbol{y}^n). \tag{21}$$

The initial residual delivered by (21) is $\boldsymbol{r}^n = \frac{(\Delta t)^3}{4}A^2 \boldsymbol{f}(t_{n+1/2}, \boldsymbol{y}^n)$, or, and for linear inhomogeneous system with $\boldsymbol{f}(t, \boldsymbol{y}) = A(t)\boldsymbol{y} + \boldsymbol{g}(t)$, we obtain

$$\boldsymbol{r}^n = \frac{(\Delta t)^3}{4}A^2(A\boldsymbol{y}^n + \boldsymbol{g}^{n+1/2}). \tag{22}$$

In a similar way as for the Euler Backward scheme, we obtain after some manipulations, with $\tilde{A} = I - \frac{\Delta t}{2}A$:

$$\begin{aligned}\boldsymbol{y}^{n+1} = {} & \tilde{A}^{-1}\left(I + \frac{\Delta t}{2}A - 2(\frac{\Delta t}{2}A)^3 R_k^{(n)}(\tilde{A})\right)\boldsymbol{y}^n \\ & + \Delta t \tilde{A}^{-1}\left(I - (\frac{\Delta t}{2}A)^2 R_k^{(n)}(\tilde{A})\right)\boldsymbol{g}^{n+1/2}.\end{aligned}$$

Assuming that A can be diagonalized, and has real negative eigenvalues, we get for the components of the transformed solutions $z^n = z_j^n$:

$$z^{n+1} = \frac{1 + \mu - 2\mu^3 R_k^{(n)}(1-\mu)}{1-\mu} z^n + \Delta t \frac{1 - \mu^2 R_k^{(n)}(1-\mu)}{1-\mu}\hat{g}^{n+1/2},$$

with $\mu = \frac{1}{2}\Delta t \lambda_j$.
The scheme is stable if for all time steps:

$$\left|\frac{1 + \mu - 2\mu^3 R_k(1-\mu)}{1-\mu}\right| \leqslant 1. \tag{23}$$

Following the same heuristic arguments as in the previous section, we conclude that the polynomial

$$Q(\mu) = 2\mu^3 R_k(1-\mu)$$

has in average almost the minimax property. This leads to the bound

$$-2.375 \leqslant \eta_1 \qquad (<0), \tag{24}$$

where η_1 is the largest among all η_i such that $R_k^{(n)}(1-\eta_i) = 0$.

4 GMRES iterations and stability control

We assume that the minimal residual iterations are carried out with GMRES [13, 1]. The iterations are applied to the linear system with matrix $\tilde{A}$ (for the Euler Backward scheme, $\tilde{A} = I - \Delta t A$). In k steps of GMRES an orthogonal basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\}$ for the Krylov subspace span$\{\boldsymbol{r}^n, \tilde{A}\boldsymbol{r}^n, \ldots, \tilde{A}^{k-1}\boldsymbol{r}^n\}$ is built up. We take these vectors $\boldsymbol{v}_i$ as the columns of the matrix V_k. GMRES constructs a small $(k+1) \times k$ upper Hessenberg system

$$\underline{\tilde{H}} u = b, \qquad b = (\|\boldsymbol{r}^n\|_2, 0, \ldots, 0)^T \in \mathbb{R}^{k+1}, \tag{25}$$

that is solved in the least squares sense, where $\underline{\tilde{H}} = V_{k+1}^* \tilde{A} V_k$.
The construction of V_{k+1} and $\underline{\tilde{H}}$ requires k matrix–vector multiplications with $\tilde{A}$ and $k(k+1)/2 + k$ inner products. The Jacobian matrix is not needed in explicit form, only its action on a vector is required. For instance, this may be approximated directly by a Frechet derivative in the direction of the vector on which the Jacobian acts. The MRAI algorithm is explicit and allows straightforward parallelization.

During the time-stepping process, the stability of MRAI can easily be checked with conditions like those inferred by (15) and (24), and, if necessary, the step size can be adjusted. We note that $\underline{\tilde{H}}$ depends on Δt as $\underline{\tilde{H}} = I - \Delta t \underline{H}$, where elements of $\underline{H}$ do not depend on Δt. The required value η_1 can be computed from the small upper Hessenberg system, since the $1 - \eta_i$ are the eigenvalues of the matrix

$$\tilde{H}^{-*}(\underline{\tilde{H}}^* \underline{\tilde{H}}),$$

where $\tilde{H}$ is the $k \times k$ upper part of $\underline{\tilde{H}}$. The values η_i are the harmonic Ritz values of $\tilde{A}$ [12].
The conditions (15) and (24) have been derived for a negative real spectrum of A. It turned out experimentally that they may also be used when the spectrum of A is contained in the left complex half-plane. Instead of η_1, we then use its real part.
For the simplified situation where $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{y})$, we have collected the major elements of two MRAI scheme, based on Euler Backward and Trapezoidal rule, in Tables 1 and 2. These allow adjustment of the step size for almost no extra price, when the Krylov basis has already been computed.

5 Numerical experiments

Numerical experiments with the MRAI approach are presented for two different situations. In the first one, an MRAI scheme is used in a physical simulation code VAC [15]. In the second case, we demonstrate that the MRAI approach can be successfully employed in an ODE code.

Table 1: First order MRAI based on Euler Backward

1.	$\boldsymbol{y}^n$ is given, $\boldsymbol{f}^n := \boldsymbol{f}(\boldsymbol{y}^n)$ $\boldsymbol{r}^n := A\boldsymbol{f}^n$, compute b in (25)	Function evaluation Jacobian action
2.	Modified Gram-Schmidt $\Rightarrow$ matrices V_{k+1} and $\underline{\tilde{H}}$	k Jacobian actions, $k+1$ vectors to store
3.	Choose Δt (cf. (15)), $b := (\Delta t)^2 b$, solve the least-square problem (25)	$\mathcal{O}(k^3)$ operations
4.	Starting vector: $\boldsymbol{y}_P^{n+1} := \boldsymbol{y}^n + \Delta t \boldsymbol{f}^n$ MRAI step: $\boldsymbol{y}^{n+1} := \boldsymbol{y}_P^{n+1} + V_k u$	 k vector updates

Table 2: Second order MRAI based on Trapezoidal rule

1.	$\boldsymbol{y}^n$ is given, $\boldsymbol{f}^n := \boldsymbol{f}(\boldsymbol{y}^n)$ $\boldsymbol{p}^n := A\boldsymbol{f}^n$, $\boldsymbol{r}^n := \frac{1}{4}A\boldsymbol{p}^n$, compute b in (25)	Function evaluation 2 Jacobian actions
2.	Modified Gram-Schmidt $\Rightarrow$ matrices V_{k+1} and $\underline{\tilde{H}}$	k Jacobian actions, $k+1$ vectors to store
3.	Choose Δt (cf. (24)), $b := (\Delta t)^3 b$, solve the least-square problem (25)	$\mathcal{O}(k^3)$ operations
4.	Starting vector: $\boldsymbol{y}_P^{n+1} := \boldsymbol{y}^n + \Delta t \boldsymbol{f}^n + \frac{(\Delta t)^2}{2}\boldsymbol{p}^n$ MRAI step: $\boldsymbol{y}^{n+1} := \boldsymbol{y}_P^{n+1} + V_k u$	 k vector updates

5.1 MRAI time stepping in the magnetohydrodynamical simulation

In this section we describe numerical experiments made with the MRAI approach in an magnetohydrodynamical (MHD) simulation code VAC (Versatile Advection Code[1]) [15], [9].

The numerical results presented here, together with others, were reported in [16], where performance of several implicit and explicit schemes are compared on problems typical for astronomical research. Here, we discuss in detail behavior of the MRAI scheme for one of the model problems to illustrate our results from the section 3.

This model problem is a 2D MHD problem which shows formation of a steady bow shock occuring in a super-fast flow around a perfectly conducting cylinder. Such a problem is of interest for simulation of the bow shock around planets in the solar wind. For more details on the model problem description see [16].

In VAC, the equations are discretized spatially by high-resolution shock-capturing finite difference schemes [17]. In this example, a polar 60×60 grid was used, and the discretization resulted in a system of $N = 21600$ nonlinear ODE's. The steady-state solution for this problem is to be obtained by the a time stepping integration. The time-stepping process is performed until the relative difference in solution becomes sufficiently small. Because the step size Δt can be much larger than in time-accurate computations, this way to get the steady state solution is sometimes called pseudo time-stepping.

With the simple Forward Euler explicit scheme, the steady solution can be obtained in 1710 time steps which requires approximately 3120 seconds of CPU time on the Sun Sparcserver 1000E.

As an alternative, an implicit time stepping could be applied. Because of prohibitively high storage requirements, direct linear solves can not be used. For problems of this type iterative solvers work satisfactory only with a powerful preconditioning as e.g. Modified (Relaxed) Block ILU [19], [16]. This again leads to high storage needs which are hardly possible to satisfy for finer grids or in 3D case. Although it is not fare to compare performance of such fully implicit scheme with the performance of explicit MRAI scheme, we report that for this 2D example the implicit scheme requires approximately 3 times less CPU time than the MRAI time stepping.

Therefore, as a cheaper alternative, the MRAI time stepping was applied. Our analysis in section 3 suggests that stability of the MRAI time stepping depends strongly on the choice of the starting vector. In particular, evaluations of the starting vector, made in successive time steps by an explicit scheme, are typically the main source of unstability of the MRAI schemes. The choice of the explicit scheme is usually determined by the order requirements (the order of an MRAI scheme is not larger that the order

[1] See URL `http://www.fys.ruu.nl/~ toth/`.

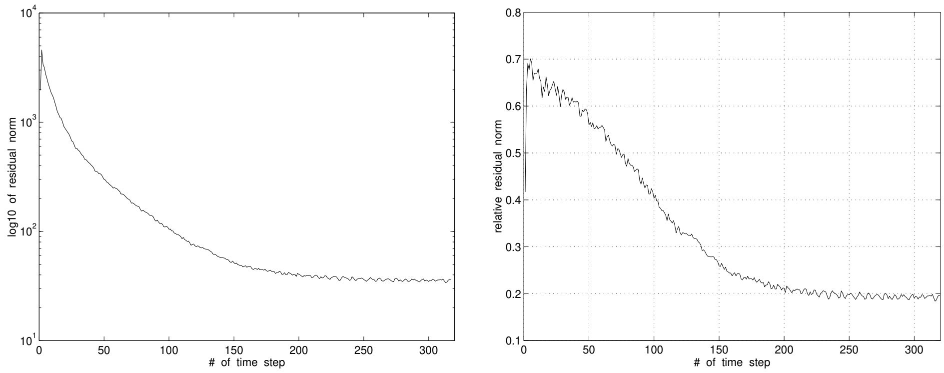

Figure 4: $\log_{10}$ of residual norm (left) and residual norm reduction (right) delivered after $k = 5$ GMRES iterations versus the time step number.

of the starting vector). In this case, since the integration process is steady state, accuracy in time is not of interest. Therefore, the starting vector was simply $\boldsymbol{y}_P^{n+1} = \boldsymbol{y}^n$. As the corrector, the linearized Euler Backward scheme was used.

It is important for the overall efficiency that this MRAI scheme allows the flexible cheap change of Δt (where Δt is adjusted after computing the Krylov basis matrix V_{k+1}). The scheme is almost identical to the one represented in Table 1, the difference is in several simplifications: $\boldsymbol{r}^n := \boldsymbol{f}^n$, $b := \Delta t b$ in step 3, and $\boldsymbol{y}_P^{n+1} := \boldsymbol{y}^n$.

Stability analysis similar to that of section 3 suggests that this MRAI scheme is nearly unconditionally stable for linear problems with negative spectrum of the Jacobian. The model problem under consideration, however, is nonlinear, with strongly nonsymmetric Jacobian, so that the step size has to be restricted.

In the stability step size control used, the value of Δt was accepted whenever η_1, the smallest in modulus element of $\{\eta_i \mid R_k(1 - \eta_i) = 0\}$, satisfied $-12.5 = \eta_1^- \leqslant \eta_1 \leqslant \eta_1^+ = -9.5$. This stability condition is similar to (15), (24). The acceptance segment $[\eta_1^-; \eta_1^+]$ was determined on a basis of numerical experiments. We found this stability condition to be reliable for the described above MRAI scheme if the norm of the skew-symmetric part of the Jacobian is of order of the norm of its symmetric part. It should be emphasized that the performance of the scheme depends only mildly on the choice of k, η_k^- and η_k^+. Our experience suggests that the choice $k = 5$ is efficient for most problems.

This variable step size implementation of MRAI($k = 5$) required 1774 seconds of CPU to get to the steady state in 317 time steps. Hence, the gain factor in CPU time with respect to the explicit scheme is at least 1.75.

An alternative way to control step size with respect to the stability could

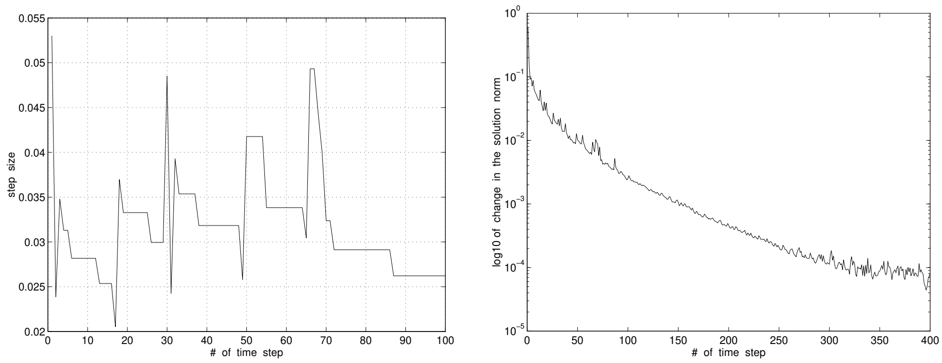

Figure 5: Step size delivered by MRAI (left) and steady state convergence history (right). (The step size did not change after first 100 time steps.)

be to restrict the step size in such a way that the residual norm $\|\boldsymbol{r}_k^n\|$ or the norm reduction $\|\boldsymbol{r}_k^n\|/\|\boldsymbol{r}^n\|$ at each time step is below certain tolerance. However, there seems to be no clear relation between the stability and the residual norm behavior. This can be seen on Figure 4, where we have plotted the residual norm $\|\boldsymbol{r}_k^n\|$ and relative norm $\|\boldsymbol{r}_k^n\|/\|\boldsymbol{r}^n\|$ as were observed during the computations with the MRAI scheme. As one can see, the residual norms varies significantly during the time stepping process, so that it is not clear what tolerance in the residual reduction criterion could have been used. On the other hand, the just described stability control based on the harmonic Ritz values provided smooth convergence to the steady state (Figure 5).

5.2 MRAI in an ODE code

In an ODE code, the step size is normally controlled automatically based on an estimation of the local error. The stability control described above is then superfluous. However, the environment typical for most ODE codes provides a fare comparison among the codes in terms of CPU time, number function evaluations and achieved accuracy. We include this section to show that our simple approach can be competitive when used in an ODE solver as well.

We incorporated the MRAI approach in the standard multistep code LSODE [6] and compared the revised code with the RKC [14] and VODPK [3] codes.

We give a short description of the methods used.

The Runge-Kutta – Chebyshev (RKC) [14, 18] code is a recent implementation of the stabilized explicit Runge-Kutta method suitable for near-stiff problems (see also [20] for a survey). The code is available from `ftp://cwi.nl/pub/bsom/rkc` and `http://www.netlib.org`.

As an implicit multistep code, the LSODE (Livermore Solver for ODE's) code has been taken. This is a general purpose stiff integrator [6], but it can work also in a non-stiff mode with Adams predictor–corrector methods. In its stiff mode, LSODE uses BLAS LU-decomposition linear solves. Banded structure of the Jacobian can be exploited by the code. Integration formulas used in LSODE and the way of the Jacobian matrix evaluation are specified by a parameter `mf` (method flag).

We incorporated the MRAI scheme in the LSODE code. Our version of LSODE code allows approximate linear solves with GMRES iterations and approximation of Jacobian action by the directional quotient as an additional option. MRAI scheme can easily be incorporated in other codes; LSODE was chosen as a widely used ODE solver, its structure and basic principles are employed in many other codes.

As it was noticed earlier, a straightforward substitution of exact linear solves by an iterative process may affect the performance of the code. For instance, after such a modification codes may deliver an actual error larger than the prescribed tolerance. An additional tuning then can be used to improve the performance [4]. We emphasize that our experiments with the modified LSODE are aimed to show only that the MRAI approach can be simply incorporated and employed in an ODE integrator. Therefore we did not change anything in the code apart from the linear solver part.

The modifications made in LSODE are activated by a new value of the LSODE method flag `mf` $= 28$. For this value, LSODE uses approximate linear solves by k GMRES iterations and the approximate "Jacobian $\times$ vector" evaluation by the directional quotient. In standard LSODE, once the Jacobian has been evaluated, it is updated only when necessary. With `mf` $= 28$, the Jacobian is always up-to-date.

The VODPK code [3] is a recently devised successor of the well known VODE code, where the direct LU linear solves are replaced by the GMRES iterative solver. As an option, the preconditioning can be provided by a user. In linear solves of VODPK, not more than 5 GMRES steps are performed. The key difference from our approach is that here the number of iterations is not fixed, the residual relative norm is controlled in the stopping criterion. If no convergence achieved within 5 GMRES steps, the step size is reduced.

All these codes have input tolerance parameters `atol` and `rtol`. It means that during the integration process the local error $\boldsymbol{e}^n$ is controlled to satisfy

$$|e_i^n| \leqslant \mathtt{rtol}|y_i^n| + \mathtt{atol},$$

where, as an option, the tolerances' parameters may be prescribed to be different for different components i. The results reported below were obtained for scalar tolerances `atol` $=$ `rtol` $=$ `tol`.

5.2.1 Example 1: 3D heat equation model problem

This test problem taken from [14] is a linear heat conduction problem in the 3D unit cube. To have an analytical solution readily available, the inhomogeneous term is specially introduced. The spatial discretization is made by central differences on the uniform grid having 39 internal nodes in each dimension. This yields system of $N = 39^3 = 59319$ equations. The integration was done for $0 \leqslant t \leqslant t_{\text{end}} = 0.7$.

For such a problem dimension, it is very expensive to use any direct Jacobian evaluation in an implicit code. Therefore, in LSODE we attempted to use an approximation to the Jacobian by using the banded evaluation ($\texttt{mf} = 25$) with the prescribed width of the band less than the actual one. This resulted in an extremely poor performance.

For the LSODE with incorporated MRAI(k) strategy, we used the method flag $\texttt{mf} = 28$ (directional quotient Jacobian evaluation), and k was set to 5. With these parameters, LSODE/MRAI required 16 N-vectors to store (10 from which are for the integrator itself and the other $k + 1 = 6$ are for the MRAI part).

Like in [14], we used VODPK with diagonal scaling preconditioning. For all the parameters in VODPK the default values were taken. The storage requirements of VODPK were 18 N-vectors to store (one vector to store for the preconditioning).

For the RKC code, all the parameters were set by default, except that we explicitly told the code an estimate for the spectral radius of the Jacobian (this is not crucial for the performance of the code, however). The RKC code requires only 4 N-vectors to store.

The results of comparative runs on the Sun Sparcserver 1000E are presented in Table 3 and on Figure 6. In the Table, columns “error”, “CPU” and “fevals/steps” contain the maximum difference in computed and exact solution at $t = t_{\text{end}}$, CPU time in seconds, number of calls to the right hand side function $\boldsymbol{f}$ and steps made, respectively.

To separate an error made in the spatial discretization from that made in the time-stepping process, the error reported in the Table is measured with respect to a reference solution computed with a stricter tolerance [14], rather than with respect to the analytical solution of the PDE.

We comment that all three codes work well for all the tolerances. The RKC code is more reliable in delivering the error compared with the tolerance. However, sometimes this code delivers error of order less than the tolerance, which means that an unnecessary work will be done to achieve a not required stricter tolerance. In opposite, the error made by VODPK may be larger than `tol`. The same, but in less extend, is true for the LSODE/MRAI code.

Summarizing, we see that for this model problem our approach works successfully and competes very well with other known techniques.

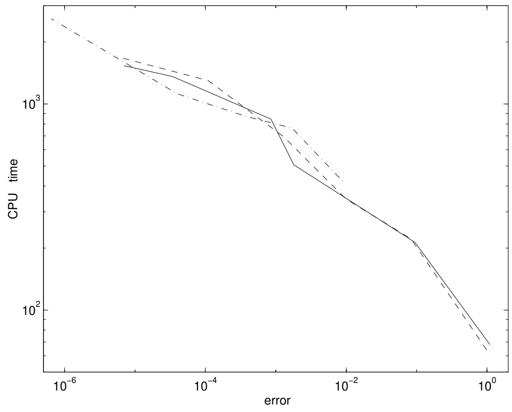

Figure 6: A log-log plot of CPU time versus error for the Example 1 (LSODE/MRAI—solid line, VODPK—dashed line, RKC—dashdotted line).

Table 3: Results for the Example 1

code	error	CPU	fevals/steps	error	CPU	fevals/steps
		$\mathtt{tol} = 10^{-1}$			$\mathtt{tol} = 10^{-4}$	
LSODE/MRAI	1.1	68	55/6	$8.5\,10^{-4}$	846	685/51
VODPK	0.99	64	46/7	$1.3\,10^{-3}$	693	480/70
RKC	$8.9\,10^{-1}$	422	402/6	$4.0\,10^{-5}$	1121	1068/57
		$\mathtt{tol} = 10^{-2}$			$\mathtt{tol} = 10^{-5}$	
LSODE/MRAI	$9.5\,10^{-2}$	213	175/12	$3.4\,10^{-5}$	1360	1087/86
VODPK	$8.3\,10^{-2}$	222	160/16	$1.1\,10^{-4}$	1298	906/115
RKC	$1.7\,10^{-3}$	764	720/15	$4.3\,10^{-6}$	1762	1760/129
		$\mathtt{tol} = 10^{-3}$			$\mathtt{tol} = 10^{-6}$	
LSODE/MRAI	$1.8\,10^{-3}$	506	403/32	$7.0\,10^{-6}$	1534	1237/98
VODPK	$1.0\,10^{-2}$	345	237/34	$6.2\,10^{-6}$	1682	1160/180
RKC	$3.7\,10^{-4}$	873	831/30	$5.1\,10^{-7}$	2588	2399/262

5.2.2 Example 2: model problem of Gear and Saad

This model problem has been adapted from the first numerical example of [5]. The original model problem in [5] is nonlinear autonomous, with the real spectrum Jacobian. We modified it in such a way that it became nonautonomous and with the complex spectrum Jacobian.

We give a brief description of the problem. Consider the set of ODE's

$$\begin{aligned} z'_{2i-1} &= \beta_i z_{2i-1} + \alpha_i z_{2i} + \gamma z_{2i-1}^2 + g_{2i-1}(t), \\ z'_{2i} &= -\alpha_i z_{2i-1} + \beta_i z_{2i} + \gamma z_{2i}^2 + g_{2i}(t), \\ &\quad i = 1, \ldots, N/2. \end{aligned}$$

It is easily checked that this system has an analytical solution

$$\tilde{z}_{2i-1}(t) = \tilde{z}_{2i}(t) = \frac{-\beta_i}{1 + c_i e^{-\beta_i t}}$$

as soon as we choose $g_{2i-1}(t) = -\alpha_i \tilde{z}_{2i}(t)$ and $g_{2i}(t) = \alpha_i \tilde{z}_{2i-1}(t)$. The constants c_i here are adjusted according to the initial values y_i^0. The change of variables

$$\boldsymbol{z} = U\boldsymbol{y}, \qquad U = I - \frac{2}{\boldsymbol{v}^T\boldsymbol{u}}\boldsymbol{u}\boldsymbol{v}^T$$

(notice that $U = U^{-1}$) leads to the system of ODE's

$$\begin{aligned} \boldsymbol{y}' &= \boldsymbol{f}(t, \boldsymbol{y}), \\ \boldsymbol{f}(t, \boldsymbol{y}) &= U\left(\Lambda U\boldsymbol{y} + \gamma (U\boldsymbol{y})^2 + \boldsymbol{g}(t)\right), \end{aligned} \tag{26}$$

where matrix Λ is block diagonal with 2×2 blocks $\begin{matrix} \beta_i & \alpha_i \\ -\alpha_i & \beta_i \end{matrix}$. The operation $(U\boldsymbol{y})^2$ for the vector $U\boldsymbol{y}$ is understood here componentwise.

This model problem has the nice property that the analytic solution $\tilde{\boldsymbol{y}}(t)$ is available readily as $\tilde{\boldsymbol{y}}(t) = U\tilde{\boldsymbol{z}}(t)$. Apart from that, the degree of nonlinearity may be controlled by the parameter γ. Also, the spectrum of the Jacobian, which is, obviously, $\beta_i \pm \alpha_i + 2\gamma z_{2i-1}(t)$, can be chosen arbitrary by setting α_i and β_i.

Here, we have taken $\gamma = 1$, $\beta_1 = -1000$, $\beta_2 = -800$, $\beta_3 = -500$, $\beta_4 = -300$, $\beta_i = -100\frac{N/2-i+1}{N/2-5}$ for $5 \leqslant i \leqslant N/2$, and $\alpha_1 = \cdots = \alpha_4 = 0$, and $\alpha_i = 0.5\beta_i$ for $5 \leqslant i \leqslant N/2$. The spectrum of the Jacobian at the $t = 0$ for these values of β_i and α_i is plotted on Figure 7. Furthermore, the matrix U was determined by setting $\boldsymbol{u} = (0, 1, \ldots, 1)^T$, $\boldsymbol{v} = (1, \ldots, 1)^T$, and the initial value was chosen as $\boldsymbol{y}^0 = U\boldsymbol{z}^0$, $\boldsymbol{z}^0 = (-1, \ldots, -1)^T$. Finally, the number of equations was taken $N = 15000$.

The Jacobian matrix is full, so that the direct linear solves, as, e.g. in LSODE code, are hardly possible. Therefore, again we use the modified LSODE/MRAI code and VODPK. The RKC code has substantial difficulties

Table 4: Results for the Example 2

code	error	CPU	fevals/steps	error	CPU	fevals/steps
		$\mathtt{tol} = 10^{-1}$			$\mathtt{tol} = 10^{-4}$	
LSODE/MRAI	0.72	21	85/15	$1.6\,10^{-4}$	121	449/78
VODPK	0.26	36	111/17	$1.0\,10^{-4}$	106	317/74
		$\mathtt{tol} = 10^{-2}$			$\mathtt{tol} = 10^{-5}$	
LSODE/MRAI	$7.3\,10^{-2}$	46	181/30	$1.7\,10^{-4}$	173	613/114
VODPK	$9.6\,10^{-3}$	70	212/40	$8.6\,10^{-6}$	147	436/101
		$\mathtt{tol} = 10^{-3}$			$\mathtt{tol} = 10^{-6}$	
LSODE/MRAI	$8.1\,10^{-3}$	70	269/49	$1.1\,10^{-6}$	213	741/150
VODPK	$1.4\,10^{-3}$	86	260/52	$1.2\,10^{-6}$	192	564/152

for this problem because the Jacobian spectrum is complex [14], and we excluded it from the comparisons.

For LSODE/MRAI, the method flag was taken $\mathtt{mf} = 28$ (this corresponds to the only affordable way of the Jacobian evaluation). The number of GMRES iterations was set to $k = 3$, so that the overall storage requirement of LSODE/MRAI were 14 N-vectors to store. In the VODPK code, all the parameters were chosen to have the default values and no preconditioning was used. In this mode, VODPK required 17 N-vectors to store.

The results of comparative runs of LSODE/MRAI and VODPK are presented in Table 4 and on Figure 7. The notations in the Table's columns are the same as in Table 3. The reported measurements of CPU time were made on SUN Sparc 4 workstation.

We observe that our approach competes very well for moderate tolerances $\mathtt{tol} \geqslant 10^{-3}$. For higher tolerances, however, its performance is comparable as well.

6 Conclusions

We have studied the effect of approximate linear solves in implicit time stepping processes. A small fixed number of GMRES iterations are performed in each linear solve. The resulted schemes can be seen as explicit stabilized schemes, which are referred to as MRAI (Minimal Residual Approximated Implicit) schemes.

The schemes under considerations are explicit, so that the stability is of concern. We propose a convenient way to display the stability and adjust

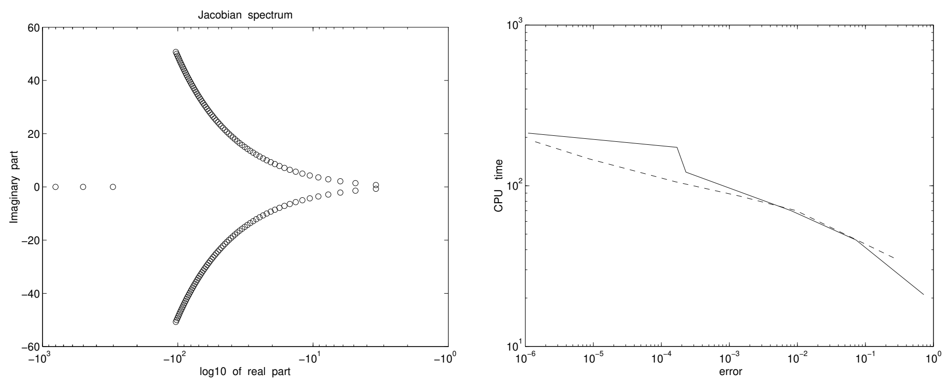

Figure 7: Jacobian spectrum at $t = 0$ (left) and a log-log plot of CPU time versus error for the Example 2 (LSODE/MRAI—solid line, VODPK—dashed line).

the step size accordingly. This is done by monitoring the harmonic Ritz values and results to an self-adaptive explicit time stepping. For autonomous systems, simple efficient implementations of MRAI up to order 2 are proposed where the step size can be changed flexibly, i.e. without recomputing the Krylov basis.

Similarly to the explicit stabilized methods and implicit methods based on iterative linear solves, MRAI approach becomes attractive when the Jacobian matrix is not easy to obtain and/or invert, or, when parallelization has to be done. However, MRAI time stepping has some distinctive features.

Usually, the convergence in linear solves is checked by displaying the residual relative norm. In many cases it is not clear what tolerance has to be taken here. In our approach, different information coming from the linear solver itself is used for the stability control.

Besides, when the residual norm is checked, the preconditioning is often indispensable to avoid the stagnation, whereas the Jacobian matrix may not be readily available. The MRAI strategy is free of this handicap.

Unlike many other explicit stabilized methods (see e.g. [20]), such as RKC, MRAI successfully copes with the complex spectrum Jacobian.

Our numerical experiments confirm these observations. They suggest that, when standard implicit schemes are too expensive and high accuracy is not required, MRAI can be very attractive. As we have shown in the experiments, the approach can be successful in both large physical simulation codes and in time accurate computations with an ODE solver.

In conclusion, we summarize that, although additional minor tuning may be necessary, the approach seems to be promising.

Acknowledgments. The first author thanks Jos van Dorsselaer for

helpful discussions. We acknowledge Ben Sommeijer (CWI) for providing us with the RKC code and his driver routine for the model problem used in section 5.2.1.
The work of Mikhail Botchev was supported by the Netherlands organization for scientific research NWO, project 95MPR04. The work of Gerard Sleijpen and Henk van der Vorst was supported in part by NWO under the same project.

References

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. A. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, PA, 1994. Available from `http://www.netlib.org/templates/`.

[2] M. A. Botchev, G. L. G. Sleijpen, and H. A. van der Vorst. Low-dimensional Krylov subspace iterations for enhancing stability of time-step integration schemes. Technical Report 1004, Department of Mathematics, Utrecht University, March 1997. Available from `http://www.math.ruu.nl/publications/`.

[3] G. D. Byrne, A. C. Hindmarsh, and P. N. Brown. *VODPK, large non-stiff or stiff ordinary differential equation initial-value problem solver.* Available from `http://www.netlib.org`, 1997.

[4] T. F. Chan and K. R. Jackson. The use of iterative linear-equation solvers in codes for large systems of stiff IVPs for ODEs. *SIAM J. Sci. Stat. Comput.*, 7(2):378–417, 1986.

[5] C. W. Gear and Y. Saad. Iterative solution of linear equations in ODE codes. *SIAM J. Sci. Stat. Comput.*, 4(4):583–601, 1983.

[6] A. C. Hindmarsh. *LSODE: Livermore solver for ordinary differential equations.* Available from `http://www.netlib.org`, 1987.

[7] M. Hochbruck and C. Lubich. On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.*, 34(5):1911–1925, October 1997.

[8] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comp.*, To appear 1997.

[9] R. Keppens, G. Tóth, M. A. Botchev, and A. van der Ploeg. Implicit and semi-implicit schemes in the Versatile Advection Code: algorithms. *Int. J. Numer. Methods in Fluids*, Submitted, 1997.

[10] V. I. Lebedev. *How to solve stiff systems of differential equations by explicit methods*, pages 45–80. CRC Press, Boca Raton, 1994.

[11] V. O. Lokutsievskii and O. V. Lokutsievskii. On numerical solution of boundary value problems for equations of parabolic type. *Sov. Math. Dokl.*, 34(3):512–516, 1987.

[12] C. C. Paige, B. N. Parlett, and H. A. van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numer. Lin. Alg. with Appl.*, 2(2):115–133, 1995.

[13] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[14] B. P. Sommeijer, L. F. Shampine, and J. G. Verwer. RKC: An explicit solver for parabolic PDEs. Technical Report MAS-R9715, CWI Amsterdam, 1997. Available from `http://www.cwi.nl/static/publications/reports/MAS-1997.html`.

[15] G. Tóth. General code for modeling MHD flows on parallel computers: Versatile Advection Code. *Astrophysical Letters & Communications*, 34:245–258, 1996.

[16] G. Tóth, R. Keppens, and M. A. Botchev. Implicit and semi-implicit schemes in the Versatile Advection Code: numerical tests. *Astronomy and Astrophysics*, To appear, 1998. Available from `http://www.math.ruu.nl/people/botchev/`.

[17] G. Tóth and D. Odstrčil. Comparison of some flux corrected transport and total variation diminishing numerical schemes for hydrodynamic and magnetohydrodynamic problems. *J. Comput. Physics*, 128:82–100, 1996.

[18] P. van der Houwen and B. P. Sommeijer. On the internal stability of explicit m-stage Runge–Kutta methods for large values of m. *Z. Angew. Math. Mech.*, 60:479–485, 1980.

[19] A. van der Ploeg, R. Keppens, and G. Tóth. Block incomplete LU-preconditioners for implicit solution of advection dominated problems. In B. Hertzberger and P. Sloot, editors, *High performance computing and networking*, Lecture Notes in Computer Science 1225, pages 421–430. Springer-Verlag, 1997.

[20] J. G. Verwer. Explicit Runge–Kutta methods for parabolic partial differential equations. *Appl. Num. Math*, 22:359–379, 1996.

[21] V. T. Zhukov. *Explicitly Iterative Difference Schemes for Parabolic Equations*, volume 4 of *Mathematical Modelling of Physical Processes*, pages 40–46. VNIIEF, 1993. In Russian.