Annales Univ. Sci. Budapest., Sect. Comp. 20 (2001) 55-70

ON FAST NUMERICAL SOLUTIONS OF SPECIAL STIFF EQUATIONS ARISING IN ASTROPHYSICS

A. and Z. Horváth (Győr, Hungary)

1. Introduction

In astrophysical calculations the interstellar matter (ISM) is often described with a system of partial differential equations (see the fundamental paper [7] and the references therein). After applying the operator splitting method (see e.g. [7], [6], [9]), in one of the splitting steps we have to solve a series of scalar initial value problems (IVPs) for ordinary differential equations (ODEs), the so-called "cooling part" of the problem (see (1) below). Note that the right hand side function of the corresponding ODE is called the cooling function.

In a typical simulation ([6]) we have to solve this cooling part 10^8-10^9 times. Classical implicit methods (implicit Euler, implicit trapezoidal rule) are used to solve this problem in the astrophysical practice, and solving the cooling part requires 20-30% of the total CPU-cost.

This fraction was lowered with a special method (called Gexp1 below) to 1-2%, see [5] and, for an application, [6]. This method is based on a *global* approximation of the cooling function in such a way that the new IVP can be solved explicitly. This results in a very robust new method.

Some test results with Gexp1 were presented in [5] without full study. In the present paper this method is studied theoretically in detail. We will show that it is convergent of first order on the relevant problems and fits our requirements at low CPU-cost. In addition, as a generalization of Gexp1, we introduce two new methods called Gexp21 and Gexp22 below which are of second order and have similarly robust as Gexp1.

In Section 4 we present some of our numerical experiments. We compare our new methods with other ones proposed for the same task, namely the implicit Euler, the BDF2, exponential Euler, exponentially fitted methods of higher order [3]. We conclude that our novel methods need significantly less CPU-time to achieve low accuracy (1-2 correct digits).

2. The IVPs to be investigated

2.1. On the original IVPs

In the cooling part of the operator splitting solution method we have to solve an IVP for e, the energy density function, concerning each discretization cell. This IVP takes the form

$$\frac{de}{dt} = -\rho^2 \cdot \Lambda\ (\rho, e) + Z \cdot \rho, \quad 0 \le t \le T, \quad e(0) = e_0, \tag{1}$$

where ρ is the density of the ISM, Z is a positive value that describes the intensity of the heating due to the radiation of the outer sources, Λ is a function which can be given only with a very complicated formula (see [4]), further T and e_0 are determined from the splitting procedure. At this stage both ρ and Z can be assumed constant. Note that in (1) we did not denote the cell-dependency of the values occured.

The components of the right hand side function of the ODE in (1) have some special properties, due to the underlying physical problem. Namely

- $\Lambda(\rho, e) > 0$ for all ρ and e;
- $\Lambda(\rho, e)$ has continuous derivatives in both variables;
- there exists $e_1 = e_1(\rho)$ such that $\Lambda\ (\rho, e)$ is proportional to e^4 if $0 < e < < e_1$;
- there exists $e_2 = e_2(\rho)$ such that $\dfrac{\partial \Lambda}{\partial e}(\rho, e) > 0$ if $0 < e < e_2$;
- there exists a unique $e_{equ.} = e_{equ.}(\rho)$ for which

$$f(e_{equ.}) = -\rho^2 \Lambda(\rho, e_{equ.}) + Z \cdot \rho = 0.$$

Note that the latter condition corresponds to the fact that there is a unique temperature (depending on the density) for which the ISM is in radiative equilibrium with the radiative field of outer sources.

Moreover, in many situations (see [6]) we have that

- for e_0, the initial value in (1), there holds $e_0 \le e_2$,

according to the task we want to model ISM at "low" temperature (less than 10^5 K).

The IVP (1) is stiff in most of our cases (i.e. most of discretization cells) because $\dfrac{\partial \Lambda}{\partial e} \cdot T$ can reach very high values. Remember that T is determined

by the other part of the outer splitting method. Another important feature of our problem is the large number of IVPs to be solved.

In virtue of these, in this paper we deal with IVPs of form

$$y'(t) = f(y(t)), \quad 0 \le t \le T, \quad y(0) = y_0, \tag{2}$$

where

(3.a) $y_0 \ge 0$ and $T > 0$ are given real numbers,

(3.b) $f \in C^1(\mathbb{R})$,

(3.c) there exists $y_e \in (0, \infty)$ such that $f(y_e) = 0$, and $f(y)$ is positive or negative if $y < y_e$ or $y > y_e$, respectively,

(3.d) $f'(y) < 0$ for all $y \in (0, \max\{y_0, y_e\})$.

2.2. The test IVPs

For the sake of perspicuity, instead of the original functions describing the radiative cooling (and heating), in our numerical experiments we shall use two families of test IVPs which are qualitatively similar to those cooling functions and fit our assumptions above.

The test IVPs have right hand side functions f_1 and f_2 with

$$f_1(y) = 1 - y^4 \cdot e^{1-y} \tag{4}$$

and

$$f_2(y) = 0.1 * (1 - y^{\alpha(y)}), \quad \text{where} \quad \alpha(y) = \begin{cases} 4, & \text{if } y < 3, \\ 4 - (y-3)/3, & \text{if } y \ge 3, \end{cases} \tag{5}$$

and we will vary T on a wide range.

Notice that f_2 does not fulfill (3.b), the differentiability condition. The reason of studying f_2 is that in practice the cooling function is often calculated by using tabulated values with piecewise linear interpolation to cut down the CPU-cost of the function evaluation (see e.g. [9]). In this way the cooling function used in the astrophysical codes is continuous with non-continuous derivative - and so is our f_2.

Our test problems have an equilibrium point $y_e = 1$. In the original problem ([7], [6]) this value depends on another parameter (density). We are able to determine it very cheaply using an interpolation array initialized once at the very beginning of the calculations. Using the initialized interpolation array, we need only a few additions and multiplications to get y_e.

3. Numerical methods

For solving the IVPs described in the previous section we need some numerical methods. Since the number of equations is very large and the underlying physical model is valid only with limited accuracy, the method must be very efficient at low accuracy, say at 1-2 correct digits. Furthermore, we require the method to be robust even in stiff situations. Here robustness also includes that the method gives approximations for $y(T)$ which tend to y_e as T goes to infinity, even if the step size is large (because T determined from the dynamic part is usually so high, that $y(T)$ is close to y_e).

There are several methods (e.g. the stiff solvers) that meet most of these requirements, but in our special case (see the conditions under (3)) it seems we can do better.

3.1. Our main method

This method was introduced in [5] motivated by the following idea: we replace f in (2) by a linear function l that approximates f *globally*, namely

$$l(y) = \frac{f(y_0)}{y_0 - y_e}(y - y_e),$$

i.e. l interpolates the points $(y_0, f(y_0))$ and $(y_e, f(y_e)) = (y_e, 0)$. We then solve the resulting linear equation exactly to get an approximation to the solution of (2). In this way we obtain a one-step method the first step of which of step size h is given by

$$y_1 = y_e + (y_0 - y_e) \exp\left(\frac{f(y_0)h}{y_0 - y_e}\right). \tag{6}$$

In the paper we call this method "global exponential method" or "Gexp1" in short.

Note that this method happens to be similar to the "exponential Euler" method (see e.g. [3] and the references therein) which arises in the same way as Gexp1, but uses a local linear approximation for f, namely the first order Taylor polynomial corresponding to the point y_0, rather than our global approximation.

We see from (6) that one step with Gexp1 requires only one function evaluation (and one exponential function calculation which is not too time consuming in modern computers). Note that we need no evaluation of the

derivative of the right hand side function, which takes usually a longer time than that of the function itself. Hence one step with Gexp1 is very cheap.

As shown in Subsection 3.2., Gexp1 is a first order method. The method can be used as a starting point for constructing higher order methods. Two second order methods derived from Gexp1 are considered in a subsequent section.

3.2. Numerical analysis of the main method

In this section we examine the method Gexp1 introduced in the previous section from the point of view of convergence. For this aim we consider the IVP. Under the assumptions (3) there exists a unique continuous function λ for which

$$f(y) = \lambda(y)(y - y_e) \quad \text{for all } y \in \mathbb{R} \tag{7}$$

holds. Note that λ is strictly negative on $\mathbb{R}\backslash\{y_e\}$.

Using the representation of (7) for f, the $n+1$st step of Gexp1 for (2) with step size $h = h_n$ reads

$$y_{n+1} = y_e + \exp(\lambda(y_n)h)(y_n - y_e) \quad \text{for } n \geq 0, \tag{8}$$

where y_{n+1} is regarded as an approximation to y at $t_{n+1} = t_n + h$ ($t_0 = 0$).

Let us investigate first the truncation error of the method. If we replace in (8) y_n and y_{n+1} by the exact solution values $y(t_n)$ and $y(t_{n+1})$, respectively, we obtain a defect $\Delta_h(t_n)$ given by

$$y(t_{n+1}) = y_e + \exp(\lambda(y(t_n))h)(y(t_n) - y_e) + \Delta_h(t_n). \tag{9}$$

We shall give an approximation to the defect in terms of the exact solution rather than that of the right hand side function of (2).

Lemma 1. *Let y be the solution of the IVP (2) where f has the properties given in (3). Then $\Delta_h(t_n) = O(h^2)$ for the defect defined in (9).*

Proof. We obtain from (7) and (2) and by employing twice a Taylor expansion that

(10)

$$\begin{aligned}\Delta_h(t_n) = y(t_n) + y'(t_n)h + \frac{h^2}{2}y''(t_n + \theta_1 h) - y_e - \\ - \left(1 + \lambda(y(t_n))h + \frac{h^2}{2}\lambda(y(t_n))^2 \exp(\theta_2\lambda(y(t_n))h)\right)(y(t_n) - y_e) = \\ = \frac{h^2}{2}(y''(t_n + \theta_1 h) - \exp(\theta_2\lambda(y(t_n))h)y'(t_n)\lambda(y(t_n)))\end{aligned}$$

with some constants $\theta_1, \theta_2 \in (0,1)$. By taking into account that $\lambda(t) \leq 0$ and $y'(t)\lambda(y(t)) = \dfrac{(y'(t))^2}{y(t)-y_e}$ for all $t > 0$, we obtain the desired result.

Next we investigate the stability of Gexp1. It is clear that the method is exact for linear autonomous problems with constant coefficients, even in the case of complex coefficients. Hence the method is A-stable.

For nonlinear stability investigations let us introduce the function

$$\Phi(z,h) = y_e + \exp(\lambda(z)h)(z - y_e), \tag{11}$$

then (8) can be written as

$$y_{n+1} = \Phi(y_n, h) \quad \text{for } n \geq 0. \tag{12}$$

For the stability we use the following definition.

Definition 1. We call the one-step method given by (12) conditionally contractive if for all VIP (2) with f having properties (3) there exists a closed, bounded interval K and $H > 0$ such that $y(t) \in K$ for all $t \geq 0, \Phi(.,h)$ maps K into K for all $0 < h \leq H$, and if, moreover, Φ satisfies

$$|\Phi(z_1,h) - \Phi(z_2,h)| \leq |z_1 - z_2| \quad \text{for all } z_1, z_2 \in K,\ 0 < h \leq H. \tag{13}$$

Lemma 2. *Gexp1 is conditionally contractive.*

Proof. Consider an IVP of form (2) with f that has the properties (3) and let K be the interval spanned by y_0 and y_e.

It is clear by (3.c) and (3.d) that $y(t) \in K$ for all $t \geq 0$. Further, by taking into account (8) and the nonpositivity of λ, we observe that $\Phi(.,h)$ maps K into K for all $h \geq 0$.

By employing the mean value theorem, we obtain that (13) is equivalent to the statement

$$\left|\frac{\partial \Phi}{\partial z}(z,h)\right| \leq 1 \quad \text{for all } z \in \text{ int}(K),\ 0 < h \leq H. \tag{14}$$

It follows from (7) and (11) by direct calculation that

$$\frac{\partial \Phi}{\partial z}(z,h) = (1 + h(f'(z) - \lambda(z)))\exp(\lambda(z)h). \tag{15}$$

The Taylor expansion of $\dfrac{\partial \Phi}{\partial z}$ with respect to h gives that $\dfrac{\partial \Phi}{\partial z}(z,h) = 1 + f'(z)h + O(h^2)$ for each fixed z, hence, by employing (3.d), there exists

$H(z) > 0$ for each $z \in K$ such that $\left|\dfrac{\partial \Phi}{\partial z}(z,h)\right| \leq 1$ for all $0 < h \leq H(z)$. Then we can see immediately that (13) is fulfilled with $H = \min\limits_{z \in K} H(z) > 0$ as well, which means the conditional contractivity of Gexp1.

Remark 1. Under special conditions much more can be proved with regard to stability of Gexp1. For example, if K is an arbitrary interval of $\mathbb{R}$ and the relative error of f' and λ is at most 1 on K, i.e. $|r(z)| \leq 1$ for all $z \in K$, where $r := \dfrac{f'(z) - \lambda(z)}{\lambda(z)}$, then (14) holds with $H = \infty$. Indeed, from (15) we have for all $z \in K$ and $0 < h$ that

$$\left|\frac{\partial \Phi}{\partial z}\right| = |(1 + h\lambda(z)r(z))\exp(\lambda(z)h)| \leq \exp(-\lambda(z)h|r(z)|)\exp(\lambda(z)h) =$$
$$= \exp(\lambda(z)h(1 - |r(z)|)) \leq 1,$$

due to $\lambda(z) \leq 0$ for all z.

Remark 2. Under the conditions in (3) the sets $[0, y_e]$ and $[y_e, \infty)$ are positively invariant for the flow generated by the ODE of (2). If follows from the arguments given in the proof of Lemma 2 that the same statement holds for the discrete flow generated by Gexp1 with arbitrary $h > 0$. This implies that no oscillations around the equilibrium appear when using Gexp1.

We conclude this subsection with the following theorem on convergence and asymptotical behaviour of Gexp1.

Theorem 1. *Consider the IVP (2) where the right hand side function f has the properties (3). Then Gexp1 is convergent of order 1 on the IVP (2). Further, if y_n is the sequence generated by Gexp1 for (2) with arbitrary $h > 0$, then $\lim y_n = y_e$.*

Proof. The first statement follows directly by combining Lemma 1 and Lemma 2 (see e.g. [1] pp. 159-162 for similar arguments), while the second one immediately from (8).

3.3. Some related higher order methods

Though Gexp1 is shown to be robust and cheap, the order of convergence is only 1. In this subsection we will show two second order methods derived from Gexp1. Note that it is possible to construct other higher order methods as well, but this is outside of our interest here.

The basic idea of our first second order method called Gexp21 is to take a trial step with Gexp1 with full timestep, get y^*, and approximate the right

side of the differential equation with a linear function between y_0 and y^*. The result of Gexp21 is the exact solution of the problem corresponding to this linear approximation. Namely

$$y^* = y_e + (y_n - y_e) \exp\left(\frac{f(y_n)h}{y_n - y_e}\right), \tag{16}$$

$$m = \frac{f(y_n) - f(y^*)}{y_n - y^*}, \tag{17}$$

$$y_{n+1} = y_n + \frac{\exp(mh) - 1}{m} \cdot f(y_n). \tag{18}$$

The other second order method we consider (called Gexp22) is a Runge-Kutta like method derived from Gexp1. It can be formalized as

$$y^* = y_e + (y_n - y_e) \exp\left(\frac{f(y_n)}{y_n - y_e}\frac{h}{2}\right), \tag{19}$$

$$y_{n+1} = y_e + (y_n - y_e) \exp\left(\frac{f(y^*)}{y^* - y_e}h\right). \tag{20}$$

Now we show that these methods are indeed of second order.

Theorem 2. *Gexp21 and Gexp22 are convergent of order 2 for IVPs of form (2) if f fulfills the conditions of (3). Moreover,* $\lim y_n = y_e$ *whenever* $n \to \infty$ *with arbitrary* $h > 0$.

Proof. The proof of the second order convergence consists of the same steps as that of Gexp1: investigation of the local error and stability in the same way as it is in Lemma 1 and Lemma 2, respectively.

As to the counterpart of Lemma 1 we have to mention that concerning Gexp21 and Gexp22 we cannot give so simple formulas for Δ_h as is in (10). Lack of this we verify $\Delta_h = O(h^3)$ by using a computer algebra program.

As to the counterpart of Lemma 2, we choose the same K set which is given in the proof of Lemma 2, but in case of the Gexp21 method $\Phi(., h)$ maps K into itself only under an additional condition $h \leq \overline{H}$ (but this does not destroy the desired conditional contractivity). Namely, $y^* \in K$ for all $h > 0$ (see Lemma 2 and (6)), hence m defined in (17) can take values only from a bounded set, which is a subset of $(-\infty, 0)$, due to (3.d). Hence we see from (18) that there exists $\overline{H} > 0$ that has the desired property.

Note that the last difference between the proof of Lemma 2 and the proof of conditional contractivity of Gexp21 and Gexp22 is that lack of a simple formula for $\frac{\partial \Phi}{\partial z}$, we verify $\frac{\partial \Phi}{\partial z}(z,h) = 1 + f'(z)h + O(h^2)$ again with the help of a computer algebra program.

Note that both of these second order methods need two evaluations of f and two calculations of the exponential function. Hence Gexp21 and Gexp22 have approximately twice as large CPU-cost as Gexp1 in each step.

3.4. The methods examined in our numerical experiments

Our aim was to construct a method with low CPU-cost achieving 1-2 correct digits and a high degree of stability. These requirements restricted the choice of numerical methods for the numerical experiments.

We examined a lot of methods to find the fastest at low accuracy in our problem. In this article we will present the results of the following ones:

1. Implicit Euler-method (IE). Though this is only a first order method, it is robust, easy to implement and needs relatively low CPU-cost at small accuracy. This is the most frequently used method in the numerical codes for astrophysical simulations. (See [7], [9].)

We used the simplified Newton iteration to solve the nonlinear equation as described in [2]. The only difference in implementation of the Newton method was that we chose $z_0 = (y_e - y_0)/2$ as the first guess, instead of $z_0 = 0$. Our numerical experiments showed that with $z_0 = (y_e - y_0)/2$ the simplified Newton iteration was always successful, but with $z_0 = 0$ we often got diverging iterations.

2. BDF2 method (BDF2). This second-order implicit method is commonly used to solve stiff equations.

Our implementation is based on [8]. We tested BDF2 both with and without automatic step-size control. The numerical experiments showed that the step-size control is economic even at low precisions (i.e. with $TOL = 10^{-1}$ in the notation of [8]), which is not self-evident according to the considerations above. On the other hand, the BDF2 method produced very poor results at high values of T without step-size control.

Therefore we will show only the results of BDF2 with step-size control.

3. Exponential Euler method (Exp2). Although this is an explicit method it is sometimes used for integrating stiff systems of equations. (See [3] and the references therein.)

The reason why we studied Exp2 is that in our case one step with Exp2 needs only one evaluation of the right hand side, one calculation for

its derivative and one evaluation of the scalar exponential function. In this way the CPU-cost of one step is significantly lower than the CPU-cost of an implicit method (like IE, BDF2).

4. 4-th order exponential Runge-Kutta method (Exp4). This method is described in [3]. We implemented it in the same way as in [3], both with and without adaptive step-size control.

5. Global exponential method (Gexp1). It is our main method. (See Section 3.1.)

We will show that at low accuracy (at 1-2 significant digits) it is the fastest method.

6. Corrected global exponential method (Gexp21); see Section 3.3.

7. Our third global exponential method Gexp22; see Section 3.3.

We studied some other methods, too. For example we implemented the 3rd order exponential Runge-Kutta method described in [3], and the various exponential Runge-Kutta methods with inaccurate Jacobian. We also tried the classical implicit trapezoid and midpoint methods. All of these methods proved to be less economic than the solvers mentioned above, hence we do not present their results.

We did not study higher order implicit methods (e.g. RADAU5), because they need a lot of function evaluations in every step, and one step with such a method is very expensive. Although these high-order methods must be better than the methods mentioned above at high accuracy ($SCD \geq 5$, see (22) below for the definition of SCD), at low accuracy they are not economic. For example with Gexp21 we could reach at relative error of 10^{-2} using 6 evaluations of the cooling function and a few multiplications and additions, while a 5th order method needs at least 6 function evaluations and a lot of auxiliary calculations to get the correct step-size in every step.

4. Numerical experiments

In this section we will present some graphs about numerical tests with the discussed methods.

In the graphs on Fig. 1 and 2 we present the CPU time[1] as a function of significantly corrected digits. Namely: we performed test calculations with

[1] The actual unit of CPU time is 10^{-6}s. Though it depends on the specific machine, and this value does not influe the differences of the methods.

different final times, i.e. for $T = 0.1,\ 0.2,\ 0.5,\ 1.0,\ 2.0,\ 5.0$. The reason of this wide-range calculations is that in our case we need a method which is stable and moderately accurate at very small and very high values of integration times, according to the expectations above. Calculations corresponding to different values of T are shown in separate graphs.

On each graph the horizontal axes show the significantly correct digits (SCDs). To obtain these results we took the following set of initial values: $Y = \{0.5, 1.3, 2.1, 2.9, 3.7\}$. The relative error we used can be formalized as

$$R(y_0, T) = \left| \frac{y(T) - y^*(T)}{y(T)} \right|, \tag{21}$$

where $y(T)$ is the exact solution and $y^*(T)$ is the value calculated with the actually discussed method.

Now we can give the formula of SCD:

$$SCD = -\log_{10} \sqrt{\frac{1}{N_y} \sum_{y_0 \in Y} R(y_0, T)^2}, \tag{22}$$

where N_y is the number of elements in Y set (actually: $N_y = 5$).

The vertical axes show the decimal logarithm of the mean square CPU-times.

The curves on the graphs arose in the following way:

- In the case of the methods with fixed time steps the dots represent the results according to $N = 1, 2, 4, 8, 16, 32, 64, 128$ number of equidistant time step.

- In the case of methods with step size control the subsequent dots correspond the results with $TOL = 0.2/2^{2i}$ for $i = 0, 1, 2, 3, 4, 5, 6, 7$, respectively.

We have the following conclusions:

1. The curves show different gradient according to the order of the method, usually. The exceptions can be found at low precision.

2. The Exp2 method shows a negative value of SCD on some graphs which is unacceptable. The reasons of this behavior is that we did not implement step-size control in this case. (In the cases of BDF2 and Exp4 without step-size control we did the same observations, too. Hence we do not present their results.)

3. In some cases the IE gives negative value of SCD, too.

4. At low precision the new method Gexp1 is the most economic, it needs the least CPU-time at $SCD = 1.5$ (which corresponds to relative error of 3% approximately).

Gexp21 is quite good at this region, too. Gexp22 produces very good results in the case of f_1 test function, but in the second test problem it becomes less accurate. The reason of this is that the derivative of f_2 is not continuous at one point in the analyzed domain.

5. At moderate precision ($1.5 < SCD < 3$) the most economic method is Gexp21 or Gexp22 depending on the time step and the test function. (In some small regions the method Gexp1 is the winner with a small difference between it and Gexp21 or Gexp22.)

In the case of "smooth" f_1 function the Gexp22 seems to be the best in this region, while in the case of f_2 it can produce less accuracy than Gexp21.

6. At high precision ($SCD \geq 5$) Exp4 is the "winner".

The algorithms were implemented in ANSI C language, and the numerical tests were performed on an Intel Pentium-based machine under Linux (RedHat 5.0) with GNU C 2.7.2.3. It is likely that our above conclusions do not depend on this hardware/software environment. There may be however small changes in the differences between implemented methods on other architectures. For example, the ratio of CPU cost of an exponential function call to the CPU-time of a multiplication can slightly influe the results.

5. Conclusions

We studied both theoretically and numerically the new method Gexp1. We proved that it is convergent of order 1 and preserves important qualitative properties of the original problem. The numerical tests showed that at low precision ($SCD = 1.0 - 1.5$) it is the most economic method in the set of examined ODE solvers.

Two new methods (Gexp21 and Gexp22) were introduced as generalisatios of Gexp1. These methods appeared to be of second order of accuracy. According to the numerical tests we conclude that these methods are the most economic at moderate precision ($1.5 < SCD < 3.0$). For problems with continuous derivatives of the right hand side Gexp22 seems to be better, otherwise Gexp21 can be preferred.

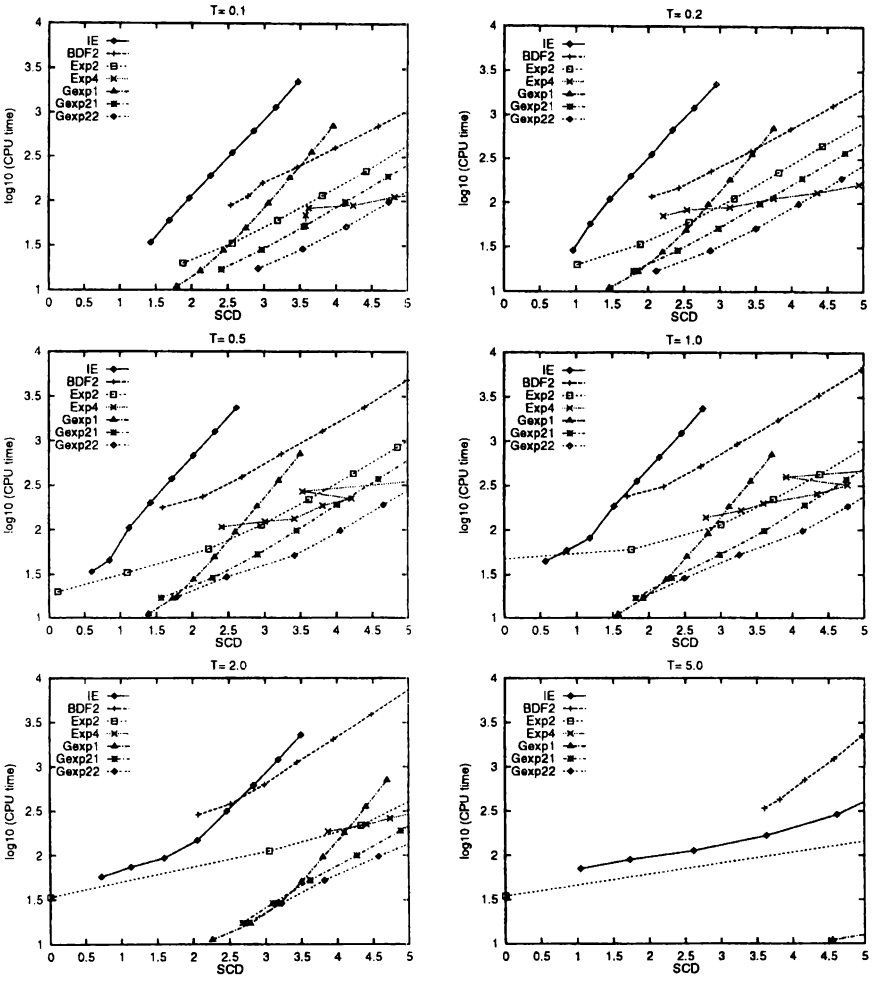

Figure 1: Numerical tests with the test function f_1. The decimal logarithm of CPU time is shown as a function of significantly correct digits (SCD, see 22) for the methods examined. The final time (T) is shown at the top of each graph. See text for details. Note that in the last graph ($T = 5.0$) Exp4, Gexp21 and Gexp22 give SCD> 5.

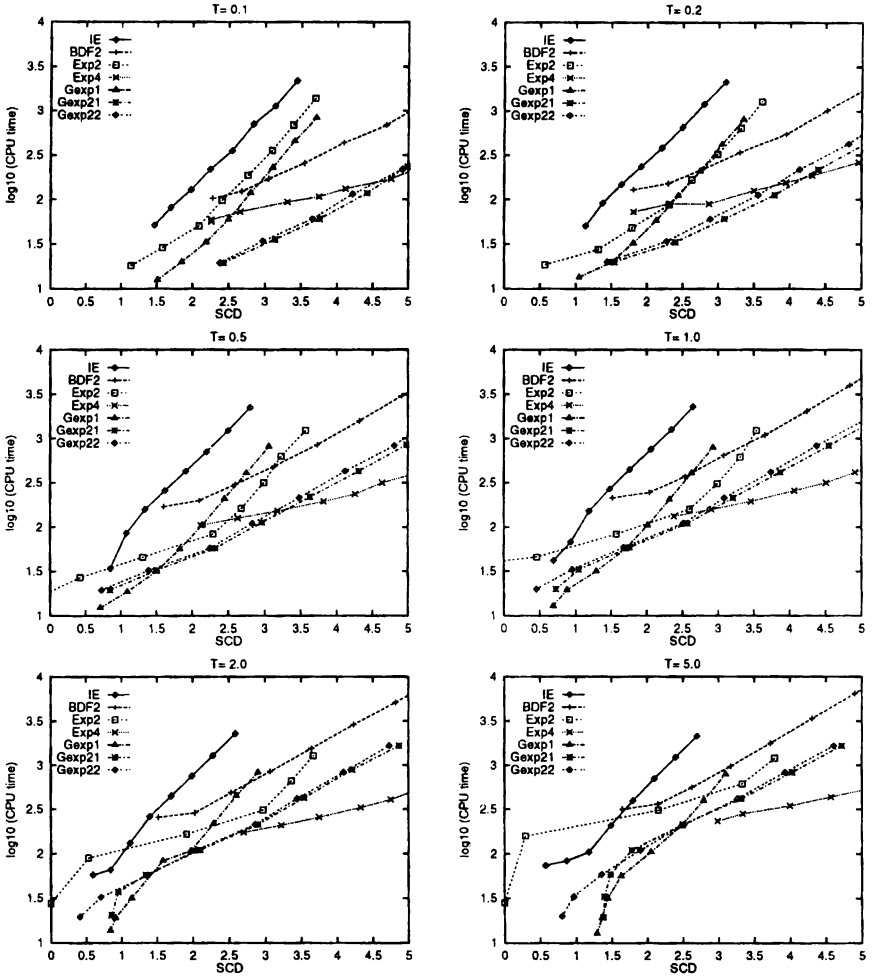

Figure 2: Numerical tests with the test function f_2. See the caption of Fig.1.

In the astrophysical problems the precision $SCD = 1.5$ is enough. To achieve it $N = 4$ steps are needed with Gexp1, and in this range it is the most economic method. Furthermore, Gexp1 even with $N = 1$ is remarkably stable. Thus Gexp1 can be proposed in astrophysical simulations of interstellar matter. To get a little bit more precision, Gexp21 or Gexp22 is recommended.

The Gexp21 method shows a very interesting behavior, it should be examined in more detail in the future. It is robust even in the case of discontinuous derivatives. To achieve $SCD = 2.0$ it can be recommended to use Gexp 21 with $N = 4$.

The Runge-Kutta like generalisation of Gexp1 is also noticeable and might be examined even with higher order generalisations later.

All the methods seem to be applicable to systems of ODEs with an attractive fixpoint. This question will be examined in a subsequent paper.

References

[1] **Hairer E., Norsett S.P. and Wanner G.,** *Solving ordinary differential equations I.,* Springer, 1993.

[2] **Hairer E. and Wanner G.,** *Solving ordinary differential equations II.,* Springer, 1991.

[3] **Hochbruck M., Lubich Ch. and Selhofer H.,** Exponential integrators for large systems of differential equations (to appear in SIAM J. Sci. Comp.)

[4] **Hollenbach D. and McKee C.** Molecule formation and infrared emission in fast interstellar shocks I., *Astrophysical Journal Supplementary Series,* **41**, 1979.

[5] **Horváth A. Jr. and Kiss Cs.,** Interstellar shock-cloud collisions - A new method for cooling, *Proceedings of May Advanced School and Work-shop on the Interaction of Stars with their Environments, Visegrád, Hungary,* eds. L.V. Tóth, M. Kun and L. Szabados, GPRINT Press, Budapest, 1997.

[6] **Horváth A. Jr. and Ziegler U.,** The influence of hydrogen molecules to the shock-cloud collisions (to appear in Astronomy and Astrophysics, 1998)

[7] **Stone J.M. and Norman M.L.,** ZEUS 2D: A radiation MHD code for astrophysical flows in two space dimensions I. The hydrodynamical algorithms and tests, *Astrophysical Journal Supplementary Series,* **80** (1992), 753-790.

[8] **Verwer J.,** Gauss-Seidel iteration for stiff ODEs from chemical kinetics, *SIAM J. Sci. Comput.,* **15** (1994), 1243-1250.

[9] **Ziegel U.,** *The role of supernova remnants to galactic dynamos,* PhD Thesis, University of Würzburg, 1995.

(Received May 22, 1998)

A. and Z. Horváth
Matematika Tanszék
Széchenyi István Főiskola
Hédervári u. 3.
H-9026 Győr, Hungary
zhorvath@kvark.szif.hu