An explicit method for solving flows of ODE

An Explicit Method for Solving Flows of ODE

B.Tasić, R.M.M.Mattheij
Department of Mathematics and Computing Science,
Eindhoven University of Technology,
PO Box 513, 5600 MB, The Netherlands

Abstract

This paper is concerned with finding numerical solutions of a flow of ODE solutions. It describes a new method, based on Euler Backward method and interpolation, for finding solutions of autonomous problems. The special aspect is that this method is explicit, and resolves the flow with similar accuracy as Euler Backward method, even when the problem is stiff. An analysis is given of both stability and accuracy and extensions to non-autonomous problems are discussed. A number of numerical examples illustrates this analysis.

1 Introduction

Implicit methods are often the method of choice for solving ODEs (Ordinary Differential Equations), like in stiff problems. In particular when they arise from semi-discretised PDEs (Partial Differential Equations), this implicitness causes the method to become very involved compared to explicit methods (see e.g. [2], [5], [6], [7], [15], [16]). The latter are often a reasonable alternative, despite the fact that they require smaller step sizes in situations where implicit methods would not (see e.g. [8], [18]). Though semidiscretised PDEs are an important source for (stiff) ODEs, there is another reason for interest. Indeed, in studying problems in fluids, one often computes a quantity like the velocity field. The position then follows from solving a simple ODE like

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n, \ n = 1, 2, 3, \tag{1.1}$$

where $\mathbf{x}$ is a point in a flow. Typically a flow is a "continuum" of solutions (see [3]). Quite often one has a bounded domain, like a (deforming) material blob (see [9], [11], [12], [13], [14], [17]). In such a case only the boundary may be of interest to describe the evolution of this blob. When trying to tackle this problem numerically, two problems

arise immediately. One is the question how to discretise the boundary at a certain time-point. The other is how to deal with an ODE, like (1.1), when **f** is not known explicitly as a function of **x**, but only as a numerical approximation of the solution **f** of the underlying equation. In particular the latter fact makes use of implicit methods virtually impossible.

In our approach we explicitly employ the fact that the flow is autonomous, which is common in many physical applications. This simple observation makes that the velocity field, in a certain space domain, is known, once it is known at a given time-level. In order to find it at other (spatial) points, we may use interpolation. In fact, we use interpolation for the positions instead. These two aspects (employing autonomy and interpolation) make that we can use implicit methods to discretise the ODE (1.1), although the method is still explicit in a way.

In this paper we will concentrate on the Euler Backward method, not only because this is the simplest one to demonstrate our algorithmic approach, but also because the error analysis is not too complex. Moreover, we will restrict ourselves to scalar problems, i.e. both x and f are scalar.

The paper is built up as follows. We first give an outline of the mathematical problem in Section 2. In this section we also describe the basic idea behind the method. Since interpolation, more in particular inverse interpolation, is an essential ingredient we briefly discuss (Lagrangian) interpolation in Section 3. The numerical solution curves found at the various time points should, of course, not "intersect" in any reasonable setting (as is e.g. implied by Lipschitz continuity). This leads to a condition which we will call "well-possedness", see Section 4. The above mentioned interpolation introduces a more complicated local error than usual. Therefore an error analysis is given in Section 5, where the local error is estimated and its global effect assessed. The most interesting aspect of our method is its stability. In Section 6 it is shown that we obtain a stability behavior very similar to that of the Euler Backward method, despite the fact that our method is explicit. There is a variety of possibilities to generalize the ideas on which this method is based and can be applied in other cases. We consider a few in Section 7. Finally, in Section 8 we give a number of numerical examples showing the quality of the method and some concluding remarks in Section 9.

2 Outline of the Method

Consider the autonomous ODE

$$\dot{x} = f(x), \tag{2.1}$$

where f is Lipschitz continuous. The notation $\dot{x}$ represents the first derivative in time and it will be used from this point onward. Let $I(t) \subset \mathbb{R}$ denote a flow of (2.1), i.e. a continuum of solutions x of (2.1), such that at time t each $x(t) \in I(t)$. In particular, we let $I(t)$ be finite, so there are boundary points, say $m(t)$ and $M(t)$, such that

$$m(t) \leq x(t) \leq M(t). \tag{2.2}$$

The full system is then formally described by

$$\begin{cases} \dot{x} &= f(x), \\ x(0) &\in I(0) = [m(0), M(0)]. \end{cases} \tag{2.3}$$

To find an approximate solution of (2.3) and thus have an approximation of $I(t)$, one can use any existing numerical method. For ease of argument and notation, we will restrict ourselves to the Euler Backward method with a fixed step size h. We denote by x^i the approximation of $x(t^i)$, with $t^i := i\,h$. Then

$$x^{i+1} = x^i + h\,f(x^{i+1}). \tag{2.4}$$

Note that, in general, the solution at t^{i+1} cannot be found analytically. This implies that one has to use some iterative method to solve (2.4). The iterative process can sometimes be a tedious and expensive task. Instead of solving x^{i+1}, from the (generally) nonlinear equation (2.4), we do the following. At time-level t^{i+1}, define the point

$$\xi^{i+1} := x^i. \tag{2.5}$$

Since (2.1) is autonomous, we clearly have

$$f(\xi^{i+1}) = f(x^i). \tag{2.6}$$

Now we can view ξ^{i+1} as the point that would have been obtained by the Euler Backward method if we would have started at ξ^i, defined by

$$\xi^{i+1} = \xi^i + h\,f(\xi^{i+1}). \tag{2.7}$$

Since ξ^{i+1} is known, we can use (2.7) to obtain the value ξ^i (at time-level t^i), i.e.

$$\xi^i := \xi^{i+1} - h\,f(\xi^{i+1}) = x^i - h\,f(x^i). \tag{2.8}$$

From (2.8) it is clear that there exists a functional dependence between points at two consecutive time-levels. In general, this dependence is unknown analytically, but at least we can use it to find approximate values. If we use interpolation, we need at least two points per unknown value. However, one should note that we are solving the flow problem (2.3), i.e. the evolution of the flow, so the variable $I(t)$. Hence we use an appropriately defined spatial discretisation of $I(t^i)$. Note that for 1-D problems only the boundaries $m(t)$ and $M(t)$ matter. However, since we would eventually like to use this method for higher dimensional problems, we need sufficiently many points to represent ("sample") $I(t^i)$. Hence we discretise the whole interval $I(t)$ somehow.

Let $\{x_k^0\}_{k=1}^n$ be a set of n ordered points in $I(0)$, such that $x_1^0 = m(0)$ and $x_n^0 = M(0)$. Then we expect $\{x_k^i\}_{k=1}^n$ to be a similarly ordered set of points, approximately in $I(t^i)$, i.e.

$$m(t^i) \doteq x_1^i < x_2^i < \ldots < x_n^i \doteq M(t^i). \tag{2.9}$$

Note that this is a consequence of well-possedness of f (following from Lipschitz continuity) and can be achieved because of consistency of the Euler Backward method (errors small enough for h small enough).

By applying (2.8) to all points x_k^i, $k = 1, \ldots, n$, we obtain a set of points $\{\xi_k^i\}_{k=1}^n$, which can be used to find solutions at time-level t^{i+1}, see Figure 1. A particular value in $I(t^i)$, can be found by interpolation. The thus described method will be referred as a *flow method*. There is no specific preference for the interpolation method, apart the requirement that the resulting approximation errors should commensurate with discretisation errors. Of course, to keep the method simple, it is reasonable to apply this interpolation either to the entire set or to some convenient subset of $\{x_k^i\}_{k=1}^n$. The interpolation issue is the main topic of the following section.

3 Interpolation

Let us rewrite (2.4) as

$$x^i = x^{i+1} - h\, f(x^{i+1}) =: F(x^{i+1}). \tag{3.1}$$

If we assume that $F(x)$ satisfies the conditions of the inverse-function theorem, i.e. in particular that $F'(x) \neq 0$, then

$$G(x) := F^{-1}(x), \tag{3.2}$$

exists, i.e. we can write

$$x^{i+1} = G(x^i). \tag{3.3}$$

Of course, the function G is unknown in general. Instead of trying to find x^{i+1} by (Newton) iteration on F, we do the following. Since we can at least find some values at time-level t^i, for which the Euler Backward values are known, we use interpolation on

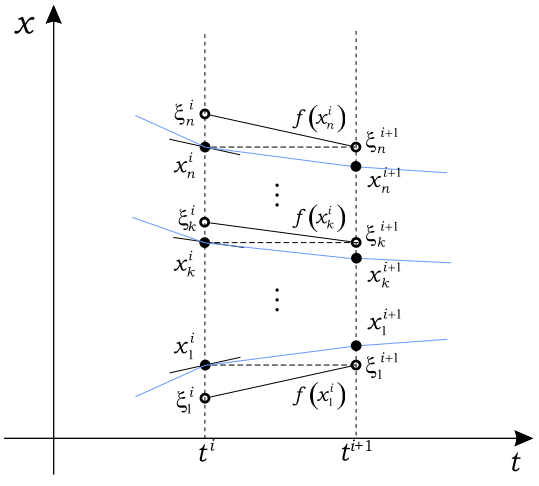


Figure 1: The principle of the flow method.

the former to obtain an approximation of G, say P, by requiring

$$x_k^{i+1} = P(x_k^i). \tag{3.4}$$

Interpolating $G(x^i)$ means that we are actually performing interpolation using the points $F(x_k^i)$. This interpolation can be local, i.e. for each point in the flow; so we do not have to use all points from $\{\xi_k^{i+1}\}_{k=1}^n$ and $\{\xi_k^i\}_{k=1}^n$. For k fixed we can use $m-1$ additional points ($m \leq n$) closest to ξ_k^i, i.e. ξ_{k+1}^i, ξ_{k-1}^i, ξ_{k+2}^i,... with corresponding values ξ_k^{i+1}, ξ_{k+1}^{i+1}, ξ_{k-1}^{i+1}, ξ_{k+2}^{i+1},... Let us denote these points by ξ_{kj}^i and ξ_{kj}^{i+1}, $j = 1, 2, \ldots, m$, respectively. Assume now that we use e.g. Lagrangian interpolation, i.e.

$$P(x_k^i) := \sum_{j=1}^{m} L_{kj}(x_k^i)\, \xi_{kj}^{i+1}, \tag{3.5}$$

$$L_{kj}(x_k^i) = \prod_{\substack{l=1 \\ l \neq j}}^{m} \frac{x_k^i - \xi_{kl}^i}{\xi_{kj}^i - \xi_{kl}^i}, \tag{3.6}$$

where the index k refers to the interpolation with respect to x_k^i. Then (3.4) becomes

$$x_k^{i+1} = P(x_k^i) = \sum_{j=1}^{m} L_{kj}(x_k^i)\, x_{kj}^i. \tag{3.7}$$

Expressed in terms of the function $G(x)$, the error of this inverse Lagrangian interpolation is given by

$$r(x) := \frac{G^{(m)}(c)}{m!}\, \omega(x), \tag{3.8}$$

where

$$\omega(x) = (x - \xi_{k1}^i) \ldots (x - \xi_{km}^i). \tag{3.9}$$

From this point onward, for simplicity, we suppress the dependence on the order m (i.e. we omit the second index), because we will mostly concentrate on linear interpolation, i.e. where $m = 2$. For this case we can use the subsequent or the previous point from the flow as a second one, i.e. x_{k+1}^i or x_{k-1}^i. Of course, the interpolation order m should be chosen high enough to guarantee sufficient accuracy; but higher order also means larger complexity.

One can express derivatives of $G(x)$ in terms of $F(x)$. Although this relation becomes increasingly complex for larger m, there is a power of $F'(x)$ in the denominator of each derivative of $G(x)$. For example

$$G'(x) = \frac{1}{F'(x)}, \quad G''(x) = -\frac{F''(x)}{(F'(x))^3}, \quad G'''(x) = -\frac{F'''(x)F'(x) - (F''(x))^2}{(F'(x))^5}, \ldots \tag{3.10}$$

By recalling (3.1), one should note that

$$F'(x) = (x - h\, f(x))' = 1 - h\, f'(x). \tag{3.11}$$

From (3.8), (3.10) and (3.11) we conclude that if

$$\left|1 - h\, f'(x)\right| \gg 1, \tag{3.12}$$

and f has bounded higher derivatives for $x(t) \in I(t)$, the interpolation error tends to zero. The condition (3.12) is typically fulfilled for stiff problems, meaning that it is reasonable to expect small interpolation errors.

Example 3.1. To show how this interpolation works out for a simple situation, consider a problem where (2.3) is linear and $m = 2$. So, let

$$\begin{cases} \dot{x} &= \lambda\, x, \\ x(0) &\in I = [m(0), M(0)], \end{cases} \tag{3.13}$$

The set $\{\xi_k^i\}_{k=1}^n$ is then defined by

$$\xi_k^i = \xi_k^{i+1} - h\, f(\xi_k^{i+1}) = x_k^i - h\, \lambda\, x_k^i. \tag{3.14}$$

From (3.5) and (3.6), we see

$$x_k^{i+1} = P(x_k^i) = \frac{x_k^i - \xi_{k+1}^i}{\xi_k^i - \xi_{k+1}^i}\, \xi_k^{i+1} + \frac{x_k^i - \xi_k^i}{\xi_{k+1}^i - \xi_k^i}\, \xi_{k+1}^{i+1}. \tag{3.15}$$

After substituting (2.5) and (3.14) into (3.15), we find

$$x_k^{i+1} = \frac{(x_k^i - x_{k+1}^i + \lambda\, h\, x_{k+1}^i)\, x_k^i - \lambda\, h\, x_k^i\, x_{k+1}^i}{(1 - \lambda\, h)\, (x_k^i - x_{k+1}^i)}, \tag{3.16}$$

leading to

$$x_k^{i+1} = \frac{x_k^i}{1 - \lambda\, h}. \tag{3.17}$$

This is exactly the same expression as one would have obtained from directly applying the Euler Backward method. This result is no surprise, of course, since linear interpolation should be exact for a linear function.

4 Well-possedness of the Flow Method

In Section 2 we assumed Lipschitz continuity of the function f. Considering the flow problem (2.3), this means that integral curves of the exact solutions $\{x_k(t^i)\}_{k=1}^n$ will not intersect (cf. [4]). We will call this well-possedness. This property should be inherited by the numerical method, i.e. it should be such that the set $\{x_k^{i+1}\}_{k=1}^n$ have the same ordering as $\{x_k^i\}_{k=1}^n$. In this section, we will seek a condition for well-possedness of

the flow method, i.e. the preservation of the ordering of points in the flow. As before, without restriction, we may assume

$$x_0^i < x_1^i < \ldots < x_n^i \Leftrightarrow \xi_0^{i+1} < \xi_1^{i+1} < \ldots < \xi_n^{i+1}, \tag{4.1}$$

No disordering of these points from the inverse time integration, is equivalent to

$$\xi_k^i < \xi_{k+1}^i. \tag{4.2}$$

By recalling (2.8), inequality (4.2) is equivalent to

$$\xi_k^{i+1} - h\, f(\xi_k^{i+1}) < \xi_{k+1}^{i+1} - h\, f(\xi_{k+1}^{i+1}), \tag{4.3}$$

or

$$x_k^i - h\, f(x_k^i) < x_{k+1}^i - h\, f(x_{k+1}^i). \tag{4.4}$$

Since $x_{k+1}^i - x_k^i > 0$, this reduces to

$$1 - h\, \frac{f(x_{k+1}^i) - f(x_k^i)}{x_{k+1}^i - x_k^i} > 0. \tag{4.5}$$

If $f(x)$ is monotonous and smooth enough for $x^i \in [x_k^i, x_{k+1}^i]$, we can approximate the second term on the left of (4.5) by the first derivative $hf'(x^i)$. For well-possedness we thus require

$$1 - h\, f'(x^i) > 0. \tag{4.6}$$

Here $f'(x)$ denotes a differentiation with respect to the argument x. The term $1 - hf'(x^i)$ in (4.6) was also encountered in the denominators of (3.10). Clearly, one should not expect that inverse interpolation will provide accurate results when this expression is close to zero. This means that, for a divergent flow, the constraint (4.6) is very strict. However, this constraint is always fulfilled for a convergent flow, i.e. for all monotonically decreasing $f(x)$. This is important for stiff problems, i.e. for problems where $h\, f'(x^i) \ll -1$. Condition (4.6) will also ensure that numerically obtained solutions of the flow do not intersect. This can be seen e.g. for linear interpolation: If we subtract two neighbouring solutions, say

$$x_{k+1}^{i+1} = x_{k+1}^i - h\, \frac{f(x_{k+1}^i)}{1 - h\, \frac{f(x_{k+1}^i) - f(x_k^i)}{x_{k+1}^i - x_k^i}}, \tag{4.7a}$$

$$x_k^{i+1} = x_k^i - h\, \frac{f(x_k^i)}{1 - h\, \frac{f(x_{k+1}^i) - f(x_k^i)}{x_{k+1}^i - x_k^i}}, \tag{4.7b}$$

then

$$\begin{aligned} x_{k+1}^{i+1} - x_k^{i+1} &= x_{k+1}^i - x_k^i - h\, \frac{f(x_{k+1}^i) - f(x_k^i)}{1 - h\, \frac{f(x_{k+1}^i) - f(x_k^i)}{x_{k+1}^i - x_k^i}}, \\ &= \frac{x_{k+1}^i - x_k^i}{1 - h\, \frac{f(x_{k+1}^i) - f(x_k^i)}{x_{k+1}^i - x_k^i}}, \end{aligned} \tag{4.8}$$

which is positive as $x_{k+1}^i - x_k^i > 0$ and assuming (4.5) is satisfied.

5 Error Analysis

In this section, we show that the local error of the flow method consists of two components. The first one is the local discretisation error arising from the Euler Backward method. The second one comes from the inverse interpolation , as shown in Section 3. We will assume that the well-possedness, described in the previous section, is satisfied. Recall (cf. e.g. [10]) that the local discretisation error of the Euler Backward method applied to (2.1) is given by

$$d_k^i := -\frac{h}{2}\,\ddot{x}_k(t^i) + O(h^2). \tag{5.1}$$

As before, we restrict ourselves to linear interpolation, for the sake of simplicity. The local error is found from substituting an exact solution into the numerical scheme, assuming the solution is exactly known at $t = t^i$. So take

$$\xi_k(t^{i+1}) := \xi_k^{i+1} = x_k(t^i) := x_k^i, \quad k = 0, 1, \ldots, n. \tag{5.2}$$

Using (3.5), with $m = 2$, one can find that the interpolation polynomial, as a function of the exact solution, is given by

$$\begin{aligned} P(x_k(t^i)) &= \frac{x_k(t^i) - \xi_k^i}{\xi_{k+1}^i - \xi_k^i}\,\xi_{k+1}^{i+1} + \frac{x_k(t^i) - \xi_{k+1}^i}{\xi_k^i - \xi_{k+1}^i}\,\xi_k^{i+1} \\ &= \frac{h\,\dot{x}_k(t^i)\,x_{k+1}(t^i) - [x_k(t^i) - x_{k+1}(t^i) + h\,\dot{x}_{k+1}(t^i)]\;x_k(t^i)}{x_{k+1}(t^i) - x_k(t^i) - h\;[\dot{x}_{k+1}(t^i) - \dot{x}_k(t^i)]} \\ &= \frac{x_k(t^i)\;[x_{k+1}(t^i) - x_k(t^i)] + h\;[\dot{x}_k(t^i)\,x_{k+1}(t^i) - x_k(t^i)\,\dot{x}_{k+1}(t^i)]}{x_{k+1}(t^i) - x_k(t^i) - h\;[\dot{x}_{k+1}(t^i) - \dot{x}_k(t^i)]}. \end{aligned} \tag{5.3}$$

After adding and subtracting $\dot{x}_k(t^i)\,x_k(t^i)$, the second term in the numerator of (5.3), can be reformulated as

$$\begin{aligned} h\left[\dot{x}_k(t^i)x_{k+1}(t^i) - x_k(t^i)\dot{x}_{k+1}(t^i)\right] &= h\left[\dot{x}_k(t^i)\left(x_{k+1}(t^i) - x_k(t^i)\right)\right. \\ &- \left. x_k(t^i)\left(\dot{x}_{k+1}(t^i) - \dot{x}_k(t^i)\right)\right]. \end{aligned} \tag{5.4}$$

Using (5.4) and the well-possedness, i.e. $x_{k+1}(t^i) - x_k(t^i) \neq 0$, (5.3) becomes

$$P(x_k(t^i)) = \frac{x_k(t^i) + h\;\left[\dot{x}_k(t^i) - x_k(t^i)\,\frac{\dot{x}_{k+1}(t^i) - \dot{x}_k(t^i)}{x_{k+1}(t^i) - x_k(t^i)}\right]}{1 - h\,\frac{\dot{x}_{k+1}(t^i) - \dot{x}_k(t^i)}{x_{k+1}(t^i) - x_k(t^i)}}. \tag{5.5}$$

We introduce the following simpler notation (note that $\dot{x}_k(t^i) = f(x_k(t^i))$)

$$\frac{\dot{x}_{k+1}(t^i) - \dot{x}_k(t^i)}{x_{k+1}(t^i) - x_k(t^i)} = \frac{f(x_{k+1}(t^i)) - f(x_k(t^i))}{x_{k+1}(t^i) - x_k(t^i)} =: \frac{\Delta f_k(t^i)}{\Delta x_k(t^i)}. \tag{5.6}$$

This leads to

$$P(x_k(t^i)) = \frac{x_k(t^i) + h\;\left[\dot{x}_k(t^i) - x_k(t^i)\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)}\right]}{1 - h\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)}} = x_k(t^i) + h\,\frac{\dot{x}_k(t^i)}{1 - h\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)}}. \tag{5.7}$$

Now the local error of the flow method is found as the residual, say $\delta(x_k(t^{i+1}), h)$, i.e. from substituting the exact solution in $x_k^{i+1} = P(x_k^i)$. We then find

$$\delta(x_k(t^{i+1}), h) = \left[x_k(t^{i+1}) - x_k(t^i) - h\,\frac{\dot{x}_k(t^i)}{1 - h\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)}}\right]/h. \tag{5.8}$$

By assuming that $\left|h\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)}\right|$ is small enough for h sufficiently small, we may use the approximation

$$\frac{1}{1 - h\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)}} = 1 + h\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)} + O(h^2), \tag{5.9}$$

to give

$$\delta(x_k(t^{i+1}), h) = \left[x_k(t^{i+1}) - x_k(t^i) - h\,\dot{x}_k(t^i)\left[1 + h\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)} + O(h^2)\right]\right]/h. \tag{5.10}$$

For a sufficiently smooth solution, one can expand the solution at the time-level t^{i+1} as

$$x_k(t^{i+1}) = x_k(t^i) + h\,\dot{x}_k(t^i) + \frac{h^2}{2}\,\ddot{x}_k(t^i) + O(h^3). \tag{5.11}$$

If we substitute this in (5.10), we have

$$\delta(x_k(t^{i+1}), h) = \frac{h}{2}\,\ddot{x}_k(t^i) - h\,\frac{\Delta f_k(t^i)}{\Delta x_k(t^i)}\,\dot{x}_k(t^i) + O(h^2). \tag{5.12}$$

Since $f(x_{k+1}(t^i)) = f\left(x_k(t^i) + \Delta x_k(t^i)\right)$, expansion around $x_k(t^i)$ leads to

$$\frac{\Delta f_k(t^i))}{\Delta x_k(t^i)} = f'(x_k(t^i)) + \frac{\Delta x_k(t^i)}{2}\,f''(x_k(t^i)) + O\left(\Delta x_k^2(t^i)\right). \tag{5.13}$$

Now the local error is found to be

$$\begin{aligned}
\delta(x_k(t^{i+1}), h) &= \frac{h}{2}\,\ddot{x}_k(t^i) - h\,f'(x_k(t^i))\,\dot{x}_k(t^i) - \frac{h}{2}\,\Delta x_k(t^i)\,f''(x_k(t^i))\,\dot{x}_k(t^i) + \\
&+ O(h)\,O\left(\Delta x_k^2(t^i)\right) + O(h^2) \\
&= -\frac{h}{2}\,\ddot{x}_k(t^i) - \frac{h}{2}\,\Delta x_k(t^i)\,f''(x_k(t^i))\,\dot{x}_k(t^i) + \\
&+ O(h)\,O\left(\Delta x_k^2(t^i)\right) + O(h^2).
\end{aligned} \tag{5.14}$$

Comparing right-hand sides of (5.14) and (5.1), we see that the expression of the local error of the flow method, contains two more terms coming from the interpolation. However, it is clear that the consistency order 1 is preserved. The second term on the right-hand side of (5.14), contains the second derivative of the function $f(x)$ with respect to the variable x. This is a consequence of the linear inverse Lagrangian interpolation, as

shown in Section 3. Clearly, if the order of the interpolation is higher, higher derivatives $f^{(m)}(x), m = 3, 4, \ldots$ are coming into play and their influence in the local error can become dominant. The expression (5.14) can be seen as a sum of the local discretisation error of the Euler Backward method and the interpolation error, viz.

$$\delta(x_k(t^{i+1}), h) = d_k^i + r_k^i, \tag{5.15}$$

where

$$r_k^i := r(x_k^i), \tag{5.16}$$

defined by (3.8).

The interpolation error affects the global error in the same way as a local error. Hence its global effect gives no real surprises. So we conclude from (5.15) that the global error is $O(h) + O(\Delta x_k(t^i))$ (see [10]). From this it is clear that interpolation may be the dominant error source and thus determine the accuracy of the method.

6 Stability

In the Example 3.1 it was shown that the flow method, applied to a (linear) test problem, produces the same results as the Euler Backward method if the interpolation error is equal to zero. This means that the stability region, defined as the part of the complex plane where the increment function

$$\psi(h\lambda) := \frac{1}{1 - h\lambda}, \tag{6.1}$$

is in modulus bounded by 1, is identical to that of the Euler Backward method. Of course, in the general (nonlinear) case we cannot say that the stability properties of the flow method are the same as those of the Euler Backward method. Hence, we need a different approach (see [1], [10]) to investigate its stability.

We can formulate our problem as a final solution of the one step method

$$x_k^{i+1} = P_{i,k}(x_k^i), \quad k \; fixed, \quad i = 0, 1, \ldots, \tag{6.2}$$

where we used the indices i and k to denote its dependence on the time level t^i and the interpolation point x_k^i. The linear interpolation polynomial is given by

$$P_{i,k}(x) = x + h \, \frac{f(x)}{1 - h \, \frac{f(x_{k+1}^i) - f(x)}{x_{k+1}^i - x}}. \tag{6.3}$$

In order to find out about the stability of the recursion (6.2), we look for the first variation. Let $\{z_k^i\}_{k \; fixed}$ denotes a small perturbation of the solution $\{x_k^i\}_{k \; fixed}$ of (6.2), such that $\{x_k^i + z_k^i\}_{k \; fixed}$ also satisfies (6.2) to the first order. Then we have

$$x_k^{i+1} + z_k^{i+1} = P_{i,k}(x_k^i + z_k^i) \doteq P_{i,k}(x_k^i) + P'_{i,k}(x_k^i) \, z_k^i, \tag{6.4}$$

where $P'_{i,k}(x^i_k)$ denotes the first derivative of (6.3) with respect to the argument x. By neglecting second and higher order terms we have

$$z^{i+1}_k = P'_{i,k}(x^i_k)\, z^i_k. \tag{6.5}$$

For stability in a nonlinear situation it is sufficient to prove contractivity of the discrete equation (6.5), i.e. this means

$$\left|P'_{i,k}(x^i_k)\right| \leq 1. \tag{6.6}$$

Carrying out the differentiation of (6.3), we have

$$P'_{i,k}(x) = 1 + h\, \frac{f'(x)\, \left(1 - h\, \frac{f(x^i_{k+1})-f(x)}{x^i_{k+1}-x}\right) + h\, f(x)\, \frac{f(x^i_{k+1})-f(x)-f'(x)\,(x^i_{k+1}-x)}{(x^i_{k+1}-x)^2}}{\left(1 - h\, \frac{f(x^i_{k+1})-f(x)}{x^i_{k+1}-x}\right)^2}. \tag{6.7}$$

After substituting x^i_k and some reordering, (6.7) becomes

$$P'_{i,k}(x^i_k) = \frac{1 - h\, \left(\frac{\Delta f^i_k}{\Delta x^i_k} - f'(x^i_k)\right)}{1 - h\, \frac{\Delta f^i_k}{\Delta x^i_k}} + h^2\, f(x^i_k)\, \frac{\frac{\frac{\Delta f^i_k}{\Delta x^i_k} - f'(x^i_k)}{\Delta x^i_k}}{\left(1 - h\, \frac{\Delta f^i_k}{\Delta x^i_k}\right)^2}. \tag{6.8}$$

For further analysis of (6.8), we first make an expansion

$$\frac{\Delta f^i_k}{\Delta x^i_k} = f'(x^i_k) + \frac{\Delta x^i_k}{2} f''(x^i_k) + \frac{(\Delta x^i_k)^2}{6} f'''(x^i_k) + O\left((\Delta x^i_k)^3\right). \tag{6.9}$$

Substituting (6.9) in (6.7), we find

$$\begin{aligned} \left|P'_1(x^i_k)\right| \;=\; & \left| \frac{1 - h\, \frac{\Delta x^i_k}{2} f''(x^i_k) + O\left((\Delta x^i_k)^2\right)}{1 - h\, f'(x^i_k) + O\left((\Delta x^i_k)^2\right)} \;+ \right. \\ & \left. +\; h^2\, f(x^i_k)\, \frac{\frac{1}{2} f''(x^i_k) + \frac{\Delta x^i_k}{6} f'''(x^i_k) + O\left((\Delta x^i_k)^3\right)}{\left(1 - h\, f'(x^i_k) + O\left((\Delta x^i_k)^2\right)\right)^2} \right|. \end{aligned} \tag{6.10}$$

The expression (6.10) needs to be analysed further. It is clear that interpolation influences the stability. If $I(t)$ is "well-sampled", i.e. if Δx^i_k is sufficiently small so as to make the last terms negligible, the condition (6.6) can be simplified to give

$$\left| \frac{1}{1 - h\, f'(x^i_k)} + \frac{h^2}{2}\, \frac{f(x^i_k)\, f''(x^i_k)}{\left(1 - h\, f'(x^i_k)\right)^2} \right| < 1. \tag{6.11}$$

Note that a similar condition for the Euler Backward method reads

$$\left| \frac{1}{1 - h\, f'(x^{i+1}_k)} \right| < 1. \tag{6.12}$$

This means that we have an additional term, It does not necessarily tend to zero for stiff problems, where $h\, f'(x_k^i) \gg 1$. Indeed, in this case (6.11) becomes

$$\left| \frac{1}{2} \frac{f(x_k^i)\, f''(x_k^i)}{(f'(x_k^i))^2} \right| < 1. \tag{6.13}$$

The latter constraint is not very severe, certainly compared to those found for explicit methods. We illustrate this in Section 8.

7 Extensions of the Flow Method

In previous sections we showed the basic idea, properties and analyses of the flow method. We restricted ourselves basically to the simplest implementation, namely linear interpolation with an equidistantly sampled initial interval $I(t^0)$. However, there are a number of extensions of the flow method. For example, we can use other interpolation techniques like Hermite interpolation or global approximation method like splines, etc. Another idea is to interpolate the points $f(\xi_k^{i+1})$ instead of ξ_k^{i+1}, using the corresponding ξ_k^i. Similarly as before, this results in obtaining an interpolation polynomial, say $p(x^i)$, which represent the approximation of $f(x_k^{i+1})$, i.e. we have

$$p(x_k^i) \doteq f(x_k^{i+1}). \tag{7.1}$$

Now we can substitute (7.1) into the Euler Backward method, obtaining

$$x_k^{i+1} = x_k^i + h\, p(x_k^i). \tag{7.2}$$

The analysis of this approach is basically the same as already given. To illustrate this, we can show that for the linear interpolation, (7.2) is equivalent to (3.5). Indeed, consider

$$p(x_k^i) = \frac{x_k^i - \xi_{k+1}^i}{\xi_k^i - \xi_{k+1}^i}\, f(\xi_k^{i+1}) + \frac{x_k^i - \xi_k^i}{\xi_{k+1}^i - \xi_k^i}\, f(\xi_{k+1}^{i+1}). \tag{7.3}$$

As before, after replacing (2.5) and some reordering, we have

$$p(x_k^i) = \frac{f(x_k^i)}{1 - h\, \frac{\Delta f_k^i}{\Delta x_k^i}}. \tag{7.4}$$

After substitution (7.4) into (7.2), we obtain

$$x_k^{i+1} = x_k^i + h\, \frac{f(x_k^i)}{1 - h\, \frac{\Delta f_k^i}{\Delta x_k^i}}, \tag{7.5}$$

which is the same expression as obtained before.

In the error analysis it was shown that the interpolation error depends not only on the time step size h, but also on the spatial step size Δx_k^i. This can be used to control the interpolation error by choosing Δx_k^i appropriately at every time-level. In other words, we can change the number of discretisation points in a flow according to accuracy requirements. Let us denote the number of points at t^i by n^i. By applying some tolerance considering the interpolation error, say $INTTOL$, we can construct an algorithm where $I(t^i)$ is resampled at every time-level. Of course, this means that we have to estimate the interpolation error for which we need higher derivatives of f. For linear interpolation, recall that the interpolation error is given by

$$r_k^i = -\frac{h}{2}\,\Delta x_k^i\; f''(x_k^i)\; f(x_k^i) + O(h)\, O\left((\Delta x_k^i)^2\right). \tag{7.6}$$

If we want to keep this below given $INTTOL$ we can do the following. Assume that the number of points in the flow is n^{i-1} at time-level t^{i-1}. By neglecting higher order terms, we can estimate $\|r_k^i\|_\infty$ for all points $\{x_k^i\}_{k=1}^{n^{i-1}}$ or just for boundary points, i.e. for $k = 1$ and $k = n^{i-1}$. Now, we choose Δx_{new}^i such that $\|r_k^i\|_\infty \leq INTTOL$ and resample $I(t^i)$ with a new number of points n^i, found from

$$n^i := \left\lfloor \frac{I(t^i)}{\Delta x_{new}^i} \right\rfloor + 1. \tag{7.7}$$

The tolerance of the interpolation error $INTTOL$ should commensurate with the discretisation error. By choosing $INTTOL$ too small, it can result in too many points in $I(t^i)$, which can significantly decrease the speed of computation. On the other hand, the accuracy will not equally be improved as the discretisation errors remain the same.

The interpolation error can also be controlled by changing the number of points m used for the interpolation. Of course, this involves higher derivatives of f and additional costs. For f smooth enough, higher order interpolation can decrease the interpolation error, but it will not affect the discretisation error. This means that the solution will not be more accurate, but only closer to one obtained by the Euler Backward method. Also, one can generate an algorithm where the number of sample points in a flow n^i, as well as the number of interpolation points m^i, are varying in time. Moreover, the number of neighbouring points $m_k^i - 1$ can differ for each point in a flow at each t^i. For all this control techniques, we need an interpolation error estimate, which can be done by standard techniques. If f is not smooth, higher order interpolation may cause numerical instabilities of the method.

The flow method can be also used for non-autonomous problems of the following type

$$\dot{x} = f(x) + g(t). \tag{7.8}$$

The principle remains the same, i.e. we first determine a set of points

$$\xi_k^i = \xi_k^{i+1} - h\, f(\xi_k^{i+1}) - h\, g(t^{i+1}) = x_k^i - h\, f(x_k^i) - h\, g(t^{i+1}), \tag{7.9}$$

and then interpolate points ξ_k^{i+1} with corresponding ξ_k^i, in order to obtain the interpolation polynomial $P(x^i)$. Since g does not depend on the spatial variable x, the additional source terms are cancelling in all denominators of $P(x^i)$ (for arbitrary m), making g appearing only as an additional term of the interpolation polynomial. To illustrate this, we can reformulate Example 3.1 by adding a source term $g(t)$ on the right-hand side of (3.13). By substituting (7.9),with $f(x) = \lambda\, x$, instead of (3.14), into (3.15), we have

$$x_k^{i+1} = \frac{(x_k^i - x_{k+1}^i + \lambda\, h\, x_{k+1}^i + h\, g(t^{i+1}))\, x_k^i - \lambda\, h\, x_k^i\, x_{k+1}^i - h\, g(t^{i+1})\, x_k^i}{(1 - \lambda\, h)\, (x_k^i - x_{k+1}^i)}, \tag{7.10}$$

leading to

$$x_k^{i+1} = \frac{x_k^i}{1 - \lambda\, h} + \frac{h\, g(t^i)}{1 - \lambda\, h}. \tag{7.11}$$

Once again, we obtain the same results as would be obtained by using the Euler Backward method.

8 Examples

In this section we will give a number of examples, to demonstrate the flow method, illustrating various properties discussed in previous sections. Thus it will be shown that if only the interpolation is done appropriately, the accuracy and the stability are approximately the same as obtained by the Euler Backward method. Moreover, we show that the flow method with linear interpolation is efficient, even for nonlinear problems.

The first example demonstrates the effect of the space-discretisation on the global error propagation. In Section 5, we saw that the interpolation error is negligible only if it is of the same order as the discretisation error.

Example 8.1. Consider the ODE

$$\begin{cases} \dot{x} &= -\arctan(10x), \\ I(0) &= [-1, 1]. \end{cases} \tag{8.1}$$

In this example, the velocity field f has large second derivatives around zero. This means that we can expect large interpolation errors (especially for linear interpolation), if $I(t^i)$ is not sampled well enough. Numerical results of the boundary points solutions together with the ∞-norm of the global error, for a small number of the grid points for the flow ($n = 3$), are shown in Table 1. To estimate the global error, we use solutions of the Euler Backward method with $h/1000$ as the "exact" ones. One should note that the accuracy of the flow method is slightly less than the one obtained by the Euler Backward method. The reason is, of course, that the interpolation error is dominant. This can easily be seen if we monitor ∞-norm of the interpolation error, i.e. the second term of the local error, given by (5.14). If we take $n = 21$ the interpolation error is smaller and the global errors of both methods are approximately of the same order, see Table 2.

Ind	Time	Solutions at boundaries		Interpolation error	Flow method	Euler Backward
i	t^i	x_1^i	x_n^i	$\|r_k^i\|_\infty$	$\|e_k^i\|_\infty$	$\|e_k^i\|_\infty$
0	0.0	-1.0000e+00	1.0000e+00	1.4421e-03	0.0000e+00	0.0000e+00
1	0.1	-8.7175e-01	8.7175e-01	1.8672e-03	1.8054e-02	8.6910e-04
2	0.2	-7.4695e-01	7.4695e-01	2.4869e-03	3.6697e-02	1.2020e-03
3	0.3	-6.2638e-01	6.2638e-01	3.4233e-03	5.6442e-02	1.7490e-03
4	0.4	-5.1113e-01	5.1113e-01	4.8914e-03	7.6732e-02	2.7087e-03
5	0.5	-4.0261e-01	4.0261e-01	7.2646e-03	9.5980e-02	4.4885e-03
6	0.6	-3.0279e-01	3.0279e-01	1.1100e-02	1.1012e-01	7.6680e-03
7	0.7	-2.1422e-01	2.1422e-01	1.6660e-02	1.1069e-01	1.1351e-02
8	0.8	-1.4007e-01	1.4007e-01	2.1262e-02	9.0668e-02	1.1082e-02
9	0.9	-8.3436e-02	8.3436e-02	1.6826e-02	5.9962e-02	7.2253e-03
10	1.0	-4.5509e-02	4.5509e-02	6.0704e-03	3.4010e-02	3.9100e-03

Table 1: The global error propagation for $h = 0.1$ and $\Delta x_k^0 = 1.0$ ($n = 3$)

Ind	Time	Solutions at boundaries		Interpolation error	Flow method	Euler Backward
i	t^i	x_1^i	x_n^i	$\|r_k^i\|_\infty$	$\|e_k^i\|_\infty$	$\|e_k^i\|_\infty$
0	0.0	-1.0000e+00	1.0000e+00	1.4421e-04	0.0000e+00	0.0000e+00
1	0.1	-8.5449e-01	8.5449e-01	2.2438e-04	1.1689e-03	1.1359e-02
2	0.2	-7.1124e-01	7.1124e-01	3.7265e-04	6.1113e-03	1.1562e-02
3	0.3	-5.7126e-01	5.7126e-01	6.7250e-04	7.5259e-03	1.1852e-02
4	0.4	-4.3630e-01	4.3630e-01	1.3458e-03	8.6893e-03	1.1914e-02
5	0.5	-3.0965e-01	3.0965e-01	3.0110e-03	8.9195e-03	1.1583e-02
6	0.6	-1.9789e-01	1.9789e-01	6.9482e-03	8.9961e-03	1.1917e-02
7	0.7	-1.1238e-01	1.1238e-01	1.1044e-02	9.9227e-03	1.1765e-02
8	0.8	-5.9678e-02	5.9678e-02	6.5057e-03	1.0274e-02	1.1082e-02
9	0.9	-3.0628e-02	3.0628e-02	1.5307e-03	7.1543e-03	7.2253e-03
10	1.0	-1.5440e-02	1.5440e-02	2.3198e-04	3.9400e-03	3.9100e-03

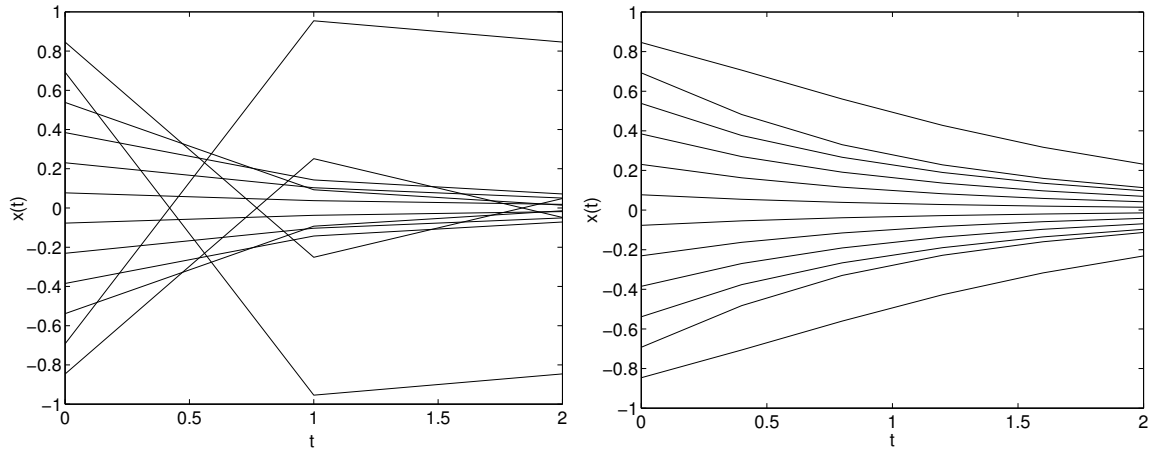
Table 2: The global error propagation for $h = 0.1$ and $\Delta x_k^0 = 0.1$ ($n = 21$)

Example 8.2. In Section 4 we formulated the condition for well-possedness of the flow method. Its usefulness is illustrated by the following example. Consider

$$\begin{cases} \dot{x} &= x\,(x-1)\,(x+1), \\ I(0) &= [-1,1]. \end{cases} \tag{8.2}$$

The right-hand side of (8.2) is the third order polynomial, which changes sign on $(-1,1)$. Note that for $x \in (-1,-\frac{1}{\sqrt{3}}) \cup (\frac{1}{\sqrt{3}},1)$ f is an increasing and for $x \in (-\frac{1}{\sqrt{3}},\frac{1}{\sqrt{3}})$ a decreas-

ing function. Since $\max_{x \in I(0)} (f'(x)) = 2$ we find that the condition for the well-possedness of the flow method (4.6) requires the time step size h to be smaller then 0.5. For larger h, we may expect the numerical integral curves to intersect for points where $f'(x) > 0$, see Figure 2a. Note that for points where $f'(x) < 0$, intersections will not occur no matter how large is h. If we use a step size $h < 0.5$, integral curves do not intersect, see Figure 2b.



(a) $h = 1.0$ (b) $h = 0.4$

Figure 2: Well-possedness of the flow method

Example 8.3. In this example we illustrate the stability properties of the flow method, as analysed in Section 6. Since stability is one of the most important reasons for using implicit methods, we show that the flow method can be successfully used for solving very stiff problems. Indeed, consider

$$\begin{cases} \dot{x} &= -10^6\, x^3, \\ I(0) &= [-1, 1]. \end{cases} \tag{8.3}$$

This is a typical example where an implicit method needs to be used. Since the stability condition of the flow method (6.11) is satisfied (in $I(t)$), we can solve (8.3) for arbitrary h. The results, obtained both with the flow method and Euler backward, for $h = 0.1$, are shown in Figure 3. One should note that the stability is preserved and that the accuracy depends on the time step only. Of course, as before, the slightly larger error of the flow method (for $t > 0.3$) is due to the interpolation error.

Example 8.4. Consider the Example (8.1) once again. It was shown that the interpolation error can play a dominant role for the accuracy of the method, if $I(t^i)$ is not sampled

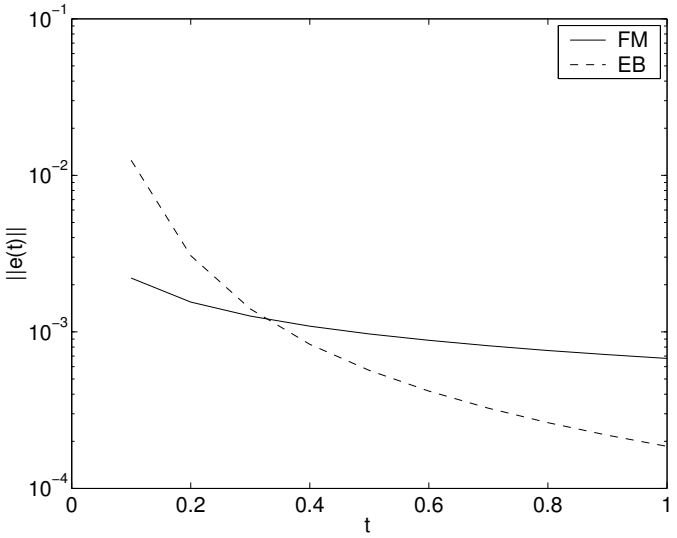


Figure 3: The global error propagation for $h = 0.1$.

well enough. Let us now solve the same problem by applying regridding as described in previous section. The interpolation error in the whole interval $I(t^i)$ is kept below a given tolerance $INTTOL$ by ensuring a sufficient number of points in the flow at every time step. The order of the discretisation error can be seen by solving this problem with the Euler Backward method. The tolerance $INTTOL$ should be of the same order, ensuring that the interpolation error commensurate with the discretisation error. In Table 3 we have given the results with $INTTOL = 10^{-3}$. One should note that the interpolation error is below $INTTOL$, which ensures that the global error of the flow method is of the same order as of the Euler Backward.

Example 8.5. In previous section, we showed that the flow method can be also used for a class of non-autonomous problems. Moreover, if f is smooth enough, the interpolation error can be reduced by using more interpolation points. In this example, we will solve a non-autonomous problem with a different number of points used for interpolation. Consider

$$\begin{cases} \dot{x} &= e^{-x} + \cos t, \\ I(0) &= [-1, 1]. \end{cases} \tag{8.4}$$

Clearly the problem (8.4) is nonlinear and non-autonomous. To obtain a proper comparison of the flow method and the Euler Backward method accuracies, we first solve the equivalent autonomous problem, i.e. we omit the source term on the right-hand side of (8.4). By comparing results obtained from the flow method, with linear interpolation ($m = 2$), with those obtained from the Euler Backward method, one can see that the interpolation error is dominant causing the global error of the flow method to be larger, see Figure 4a. This result is expected, since we are solving a highly nonlinear problem. If we now apply the flow method, again with linear interpolation, to

Ind	Time	Flow points	Interpolation error	Flow method	Euler Backward
i	t^i	n^i	$\|r_k^i\|_\infty$	$\|e_k^i\|_\infty$	$\|e_k^i\|_\infty$
0	0.0	4	9.6143e-04	0.0000e+00	0.0000e+00
1	0.1	5	9.6386e-04	3.3085e-03	8.7668e-04
2	0.2	7	8.9748e-04	5.0371e-03	1.2125e-03
3	0.3	9	9.9317e-04	6.1222e-03	1.7641e-03
4	0.4	14	9.6319e-04	6.9853e-03	2.7322e-03
5	0.5	23	9.6020e-04	7.2532e-03	4.5278e-03
6	0.6	37	9.8771e-04	6.5328e-03	7.7383e-03
7	0.7	42	9.9805e-04	4.4143e-03	1.1468e-02
8	0.8	18	9.4590e-04	2.7440e-03	1.1210e-02
9	0.9	4	9.5103e-04	2.0831e-03	7.3141e-03
10	1.0	2	4.0467e-04	1.3585e-03	3.9591e-03

Table 3: The global error propagation for $h = 0.1$ and $INTTOL = 10^{-3}$.

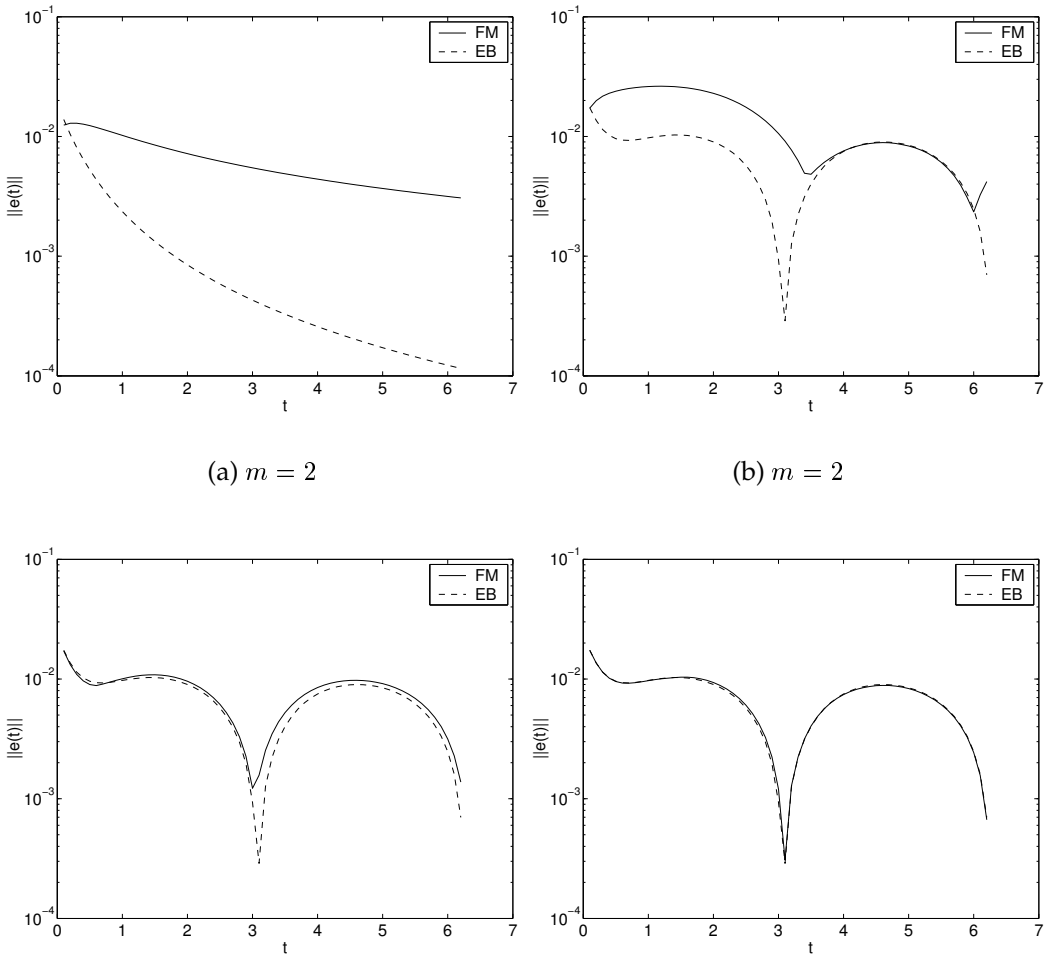
the non-autonomous (8.4), it is clear that the accuracy will be also determined by the interpolation error. This can be seen especially in areas where the discretisation error (the error of the Euler Backward method) is small, see Figure 4b. However, note that the accuracy of the flow method is still of the proper order although the problem is non-autonomous. If we increase the number of interpolation points for each point in a flow, the global error of the flow method is approaching to one of the Euler Backward method. Results for three and four points interpolation are shown on Figure 4c and Figure 4d respectively.

9 Conclusion

The flow method has been shown to have a great potential for solving autonomous flow problems, even very stiff ones. It is a, de facto, explicit method with stability properties comparable to those found from the implicit Euler method. The interpolation involved is local and does not introduce large additional costs in computation. There is a variety of ways to make the algorithm even more versatile. Despite its strong leaning on the ODE being autonomous, it can be used for non-autonomous problems as well if the time dependence appears as an additional source term only.

References

[1] J.C. Butcher. Numerical methods for ordinary differential equations in the 20th century. *Journal of Computational and Applied Mathematics*, 125:1–29, 2000.



(a) $m = 2$ (b) $m = 2$

(c) $m = 3$ (d) $m = 4$

Figure 4: The global error propagation of the autonomous (a) and the non-autonomous problem (b-d) for a different number of points used for the interpolation. Here $n = 6$ and $h = 0.1$.

[2] G.D. Byrne and A.C. Hindmarsh. Stiff ODE solvers: A review of current and coming attractions. *Journal of Computational Physics*, 70:1–62, 1987.

[3] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer-Verlag, 1997.

[4] P. Hartman. *Ordinary differential equations*. John Wiley & Sons, 1964.

[5] A.C. Hindmarsh and L.R. Petzold. Algorithms and software for Ordinary Differential Equations and Differential-Algebraic Equations, Part I: Euler methods and

error estimation. *Computers in Physics*, 9:34–41, 1995.

[6] K. Laevsky and R.M.M. Mattheij. Determining the velocity as a kinematic boundary condition in a glass pressing problem. Technical Report RANA 01-09, Eindhoven University Of Technology, 2001.

[7] D. Lanser, J.G Blom, and J.G. Verwer. Time integration of the shallow water equations in spherical geometry. *Journal of Computational Physics*, 171:373–393, 2001.

[8] V.I. Lebedev. Explicit difference schemes for solving stiff systems of ODEs and PDEs with complex spectrum. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 13:107–116, 1998.

[9] R.M.M. Mattheij and K. Laevsky. Numerical volume preservation of a divergence free fluid under symmetry. Technical Report RANA 01-11, Eindhoven University Of Technology, 2001.

[10] R.M.M. Mattheij and J. Molenaar. *Ordinary differential equations in theory and practice*. John Wiley & Sons, 1996.

[11] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[12] D. Ramsden and G. Holloway. Evolution of vesicles subject to adhesion. *Journal of Computational Physics*, 95:101–116, 1991.

[13] S.W. Rienstra and T.D. Chandra. Analytical approximations to the viscous glass flow problem in the mould-plunger pressing process, including an investigation of boundary conditions. *Journal of Engineering Mathematics*, 39:241–259, 2001.

[14] R. Rosso, A.M. Sonnet, and E.G. Virga. Evolution of vesicles subject to adhesion. *R. Soc. Lond. Proc. Ser. A Math. Phys. Eng. Sci.*, 456(1998):1523–1545, 2000.

[15] A. Sandu, F.A. Potra, V. Damian-Iordache, and G.R. Carmichael. Efficient implementation of fully implicit methods for atmospheric chemistry. *Journal of Computational Physics*, 129:101–110, 1996.

[16] R.P. Tewarson, H. Wang, J.L. Stephenson, and J.F. Jen. Efficient solution of differential equations for kidneyconcentrating mechanism analyses. *Appl. Math. Lett.*, 4:69–72, 1991.

[17] G.A.L. van de Vorst. Numerical simulation of axisymmetric viscous sintering. *Engineering Analysis with Boundary Elements*, 14:193–207, 1995.

[18] J.G. Verwer and D. Simpson. Explicit methods for stiff odes from atmospheric chemistry. *Applied Numerical Mathematics*, 18:413–430, 1995.