A new mesh selection strategy with stiffness detection for explicit Runge–Kutta methods

Francesca Mazzia a,⇑, A.M. Nagy b

a Dipartimento di Matematica, Università di Bari, Via Orabona 4, 70125 Bari, Italy
b Department of Mathematics, Benha University, 13518 Benha, Egypt

ABSTRACT

In this paper, we develop a new mesh selection strategy based on the computation of some conditioning parameters which allows to give information about the conditioning and the stiffness of the problem. The reliability of the proposed algorithm is demonstrated by some numerical experiments. We observe that “when an initial value problem is run on a computer, the results may appear plausible even if they are unreliable because of some unrecognized numerical instability” (Miller, 1967) [23]. The additional information about the behavior of the numerical solution provided by the new mesh selection algorithm are, therefore, of great interest for potential users of a numerical computer code.

1. Introduction

We consider initial value problems for systems of ordinary differential equations

\[ y' = f(t,y), \quad y(t_0) = y_0, \quad t \in [t_0, t_f], \]

where \( y_0 \) is a given vector in \( \mathbb{R}^m \).

In order to suitable choose the most efficient numerical scheme for computing a numerical solution of (1), information about the behavior of the solution are required. One of such information concerns the “stiffness” of the problem. In fact if we try to solve a stiff problem with an explicit method, the code may fails to give a solution, or may provide unreliable results.

Many researchers have attempted to find a suitable way to automatically detect stiffness when an explicit numerical schemes is used [1–5]; in practice, they detect if the stepsize is limited by stability reason. This is an important advice when we are working with explicit methods, but does not give information about the degree of stiffness of the problem to be solved. In [1] the author looks for a method that is able to recognize when the step-size is limited by stability. He used two error estimators of different orders \( err = O(h^p) \), \( err = O(h^q) \), with \( q < p \), usually \( err \gg err \), if the stepsize is limited by stability requirements and \( err \gg err \) when the stepsize is limited by accuracy requirements. The same author reports in [3], a method based on the estimation of the spectra of the Jacobian matrix that detects if the stepsize is restricted by stability. Both procedures are described in [6, p.21]. The technique based on an approximation of the dominant eigenvalue has been implemented in the code DOPRI5 [6] and used in [7,5]. The estimation of the dominant eigenvalues with Arnoldi’s methods is analyzed in [4]. Automatic selection of appropriate methods for solving stiff and nonstiff differential equation...
has been analyzed in [2,8,9]. These methods are based on the approximation of the dominant eigenvalue and/or on the estimation of the computational cost.

We report here the following sentence, quoted by [10], which summarize what is, now, considered the ‘best way’ to detect stiffness: “...Often the best way to proceed is to try one of the solvers intended for non-stiff systems. If it is unsatisfactory, the problem may be stiff. If the problem is stiff, there are effective solvers available”.

A practical definition of stiffness has been given in [11,12] and further refined in [13,14]. This definition has been used to detect stiffness for Boundary Value Problems (BVPs) and to define new hybrid mesh selection strategies based on the conditioning parameters and local error. The definition of the conditioning parameters has been given [14–16] and they measure the sensitivity of the problem to perturbations. Such parameters can be defined both for the continuous and discrete problems and for the discrete ones giving the possibility to measure how well the discrete problem approximates the continuous one. Since such parameters for the discrete problem, also depend on the chosen mesh it is possible for a fixed method to vary the mesh in order to compute a better approximation of them. Some general purpose codes for BVPs now include the computation of the conditioning parameters and hybrid mesh selection strategies based on conditioning, see, for example, the Matlab codes bvpSolve [19,20], the fortran codes twpbvpc,twpbvplc [21,22].

The mesh selection algorithm that we present in this paper is strongly related to the hybrid mesh selection algorithms for BVPs and allow the computation of the conditioning parameters. The algorithm essentially computes two solutions starting from initial conditions that are very close, and adapt the mesh in order to have a good approximation of both solution. It is possible to monitor the differences between the two solution using two different norms and this is used to give information about stability and stiffness. We observe that a similar experimental method was proposed in [23] for testing numerical stability in Initial Value Problems.

In Section 2 we recall how to analyse the conditioning of an initial value problem defined in a fixed interval and the definition of uniform stability. This definition has been used in [24] in a new algorithm to detect stiffness, here we also introduce a new definition for general non-uniformly stable problems. In Section 3 we describe the mesh selection algorithm with stiffness detection implemented in codes based on explicit Runge–Kutta methods. In Section 4 we generalize the new mesh selection strategy for non-uniformly stable problems.

The numerical tests show that the new mesh selection algorithm is able to compute a solution more accurate than the one obtained with the standard mesh selection and to give information about stability and stiffness for very difficult non linear problems used in applications [25].

2. Conditioning, stability and stiffness

Let us consider a linear initial value problem having an uniformly asymptotically stable reference solution:

\[ y' = A(t)y + q(t), \quad y(t_0) = y_0, \quad t \in [t_0, t_f] \]

(2)

The condition of uniform stability yields the definition of the stability constant

\[ \kappa_c = \sup_{t_0 < t < t_f < \infty} \| Y(t)Y^{-1}(x) \|, \]

where \( Y(t) \) is the fundamental solution of the ODE (2). We note that this stability constant is a special case of the conditioning constant for a BVPs, which for this reason, we call the conditioning constant (see [11, p.239, 12] and [26, p.100] for more details). Using \( \kappa_c \), it is possible to give a bound on the effect of perturbation \( \eta \) on the initial condition \( y_0 \). In fact, if we consider the perturbed problem:

\[ y' = A(t)y + q(t), \quad y(t_0) = y_0 + \eta, \quad t \in [t_0, t_f], \]

(3)

the difference between the solution of the perturbed and the unperturbed problem, satisfies:

\[ z' = A(t)z, \quad z(t_0) = \eta, \quad t \in [t_0, t_f], \]

(4)

and

\[ \| z \|_\infty \leq \kappa_c \| \eta \|. \]

For uniformly asymptotically stable problems we have that there exist \( \alpha > 0, \beta > 0 \) such that \( \| Y(t)Y^{-1}(x) \| \leq e^{-\beta(t-x)} \) for all \( x > t \), this means that \( \kappa_c \leq \beta \) and we say that the IVP is well-conditioned if \( \beta \) is of moderate size (see [12, p.7]).

Having fixed problem (2), we now describe one parameter which tells us if the solution varies rapidly or not in \( [t_0, t_f] \). This can be done easily by introducing the following two measures related to the solution of problem (4) with different starting values (see, e.g., [13,14]):

\[ \kappa_c(t_0, t_f, \eta) = \frac{1}{\| \eta \|_{[0, t_f]}} \max_{[t_0, t_f]} || z(t) ||, \quad \kappa_c(t_0, t_f) = \max_{\| \eta \|_{[0, t_f]} = 0} \kappa_c(t_0, t_f, \eta). \]

(5)
where \( \| \cdot \| \) is a compatible vector norm and \( \delta \) is sufficiently small; the first one study the conditioning in the infinity norm, the second one in the scaled \( l_1\)-norm. We have changed the notation, with respect to the one used in [14], giving explicitly the dependence on \( t_0, t_f \), instead only on the width of the interval \( T = t_f - t_0 \). This has been done in order to maintain the same notation needed for the general case. According to [14], we report the following definition of stiffness for uniformly asymptotically stable problems:

**Definition 1.** The initial value problem (2) is stiff in \([t_0, t_f], t_f = (t_0 + T), T > 0\) if there is a \( \delta > 0 \), such that:

\[
\sigma_e(T) = \max_{\|\eta\| > 0} \frac{K_e(t_0, t_0 + T, \eta)}{\|\eta\|} > 1.
\]

We note that \( \sigma_e(T) \) depends on the width of the interval of integration. More details about \( \sigma_e(T) \) and the relation with Definition 1 and classical definition of stiffness are reported in [14]. Since in practical applications more general problems could arise, that are no uniformly asymptotically stable, we give here a new more refined definition of stiffness for IVPs that allows to cover more general situations.

**Definition 2.** The initial value problem (1) is called stiff in the interval \([t_0, t_f]\) if

\[
\sigma^c_e(t_0, t_f) = \max_{t_0 < t < t_f} \max_{\|\eta\| > 0} \frac{K_e(t, t, \eta)}{\|\eta\|} > 1.
\]

More details about the importance of this generalized definition will be given in Section 4. In the general nonlinear case (1) if \( \hat{y} \) is the solution of the following perturbed problem:

\[
\hat{y}' = f(t, \hat{y}), \quad \hat{y}(t_0) = y_0 + \eta, \quad t \in [t_0, t_f]
\]

the difference between the solution of the perturbed and the unperturbed problem, satisfies:

\[
\zeta' = f(t, \hat{y}) - f(t, y), \quad \zeta(t_0) = \eta, \quad t \in [t_0, t_f]
\]

and the values of the conditioning parameters in (5) are computed using \( z \) solution of (9).

3. Hybrid mesh selection strategy and computation of the conditioning parameters for explicit Runge–Kutta schemes

In the following we describe the implementation of the hybrid mesh selection strategy in the MATLAB version of the code DOPRI5 even though, in principle, the algorithm could be implemented in every Runge–Kutta code. The same algorithm has been, in fact, implemented in a code based on the explicit Cash–Karp–Runge–Kutta method available in the R package deTestSet [27–30], code cashkarp with input parameter stiffness = 4, with similar results.

A general \( s \)-stage explicit Runge–Kutta method is given by

\[
y_{n+1} = y_n + h_n \sum_{i=1}^{s} b_i k_i
\]

where

\[
k_i = f(x_n + c_i h_n, g_i), \quad i = 1, \ldots, s
\]

and

\[
g_i = y_n + h_n \sum_{j=1}^{i-1} a_{ij} k_j.
\]

For the Dormand and Prince method we have \( s = 7 \) and \( c_6 = c_7 = 1 \). A numerical code computes an approximation of \( y \), solution of (1) on the grid \( \pi = (t_0, t_1, \ldots, t_N) \) with grid spacing \( h_n = t_n - t_{n-1}, n = 1, \ldots, N \). The grid is automatically computed using an estimation of the local truncation error in an attempt to produce a numerical solution that satisfy some input error tolerances. In addition to this, we would like to compute an approximation of the conditioning parameters defined in (5)–(7). To do this we also compute an approximation of \( \hat{y} \), the solution of (8), by choosing a suitable value of the perturbation \( \eta \). To compute an approximation of \( \hat{y} \) that satisfy the same input error tolerance we estimate the error for \( \hat{y} \) and we choose the stepsize accordingly. Moreover, the estimation of the conditioning parameter is based on the difference \( z = \hat{y} - y \) computed using Eq. (9) that also need to be computed with a similar accuracy.

In order to control the local error the code DOPRI5 uses, at each step \( n \), a standard relative error estimation on \( y \),

\[
e_y(h) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left( \frac{E_{ii}(h)}{\rho_i} \right)^2},
\]

where \( \rho_i = atol + \max(|y_{n,i}|, |y_{n+1,i}|)rtol \), and \( E_n(h) \) is an approximation of the local error. To have a good approximation of \( \hat{y} \), we check the error using the same strategy used for \( y \), whereas for \( z \) it is necessary to use a smaller absolute tolerance, since \( z \) is the difference between two solutions that are very close each other, so the error estimation for \( z \) is computed using the absolute tolerance equal to \( 10^{-2}atol \). This generate a new mesh selection strategy:

\[
\begin{aligned}
h_{\text{new}} &= fac \cdot h \cdot e(h)^{-1/5}, \\
e(h) &= \max(e_y(h), e_z(h)).
\end{aligned}
\]

The step is accepted if \( e(h) < 1 \). The factor \( fac \) is computed using the PI step size control \([6, p.28]\).

This allow us to approximate the conditioning parameters as follows:

\[
\begin{aligned}
\kappa_\eta(\pi) &= \frac{1}{\|\eta\|_2} \max_{i=1\ldots N} (||Z_i||_2), \\
\gamma_\eta(\pi) &= \frac{1}{\|\eta\|_2} \frac{1}{(t_N - t_0)} \sum_{i=1}^{N} h_i (||Z_i||_2 + ||Z_{i-1}||_2),
\end{aligned}
\]

where \( z_i = \hat{y}_i - y_i \), and the stiffness ratio \((6)\) as follows:

\[
\sigma_\eta(\pi) = \frac{\kappa_\eta(\pi)}{\gamma_\eta(\pi)}. \tag{13}
\]

We also monitor the relative difference between \( y \) and \( \hat{y} \) at each step \( n \) using the following formula:

\[
r_z(\pi) = \left( \frac{1}{m} \sum_{i=1}^{m} \left( \frac{y_{n,i} - \hat{y}_{n,i}}{10^{-2}atol + |y_{n,i}|rtol} \right)^2 \right)^{1/2}.
\]

In practice we compute two solutions of the same problem starting with a different initial condition. It is important to detect the choice of \( \eta \) that yields a good approximation of \( \sigma_y(T) \). After some extensive numerical experiments we choose \( \eta \) as the dominant eigenvector of \( J(t, y(t)) \), where \( J \) refers to the Jacobian matrix, with \( t \approx t_0 \). This eigenvector, in the modified version of the code DOPRI5, is computed by using the technique described in \([6]\) as follows:

\[
\eta = \xi \frac{g_7 - g_6}{||g_7 - g_6||}.
\]

and we choose the scaling factor \( \xi = rtol ||y_0||, \) if \( ||y_0|| > 0 \) and \( \xi = atol \) when \( ||y_0|| = 0 \). Moreover, to avoid a too small value of \( \xi \) we require \( \xi > 10^{-6} \cdot \text{eps} \), where \( \text{eps} \) is the machine precision. Since both \( g_7 \), and \( g_6 \) are approximation of the continuous solution at the same point \( t_0 + h_0 \), with \( h_0 \) the initial stepszie, the chosen perturbation \( \eta \) is the one naturally introduced by the numerical method used. In addition it is related to the input tolerances by the scaling factor.

For uniformly asymptotically stable problem, the stiffness detection algorithm is based on the value of \( \sigma_\eta(\pi) \). In more details, we empirically consider a problem to be stiff when \( \sigma_\eta(\pi) > 50 \) and \( r_z(\pi) \) is small, naturally this is an empirical decision.

We can have more information about the behavior of the numerical methods analyzing the two approximations of the errors \( e_y(h) \) and \( e_z(h) \). In fact if \( e_y(h) \) is smaller than \( e_z(h) \), we can deduce that the stepsize is restricted only by stability reason, moreover we can deduce that the numerical solution is unstable if \( r_z(\pi) \) and \( \kappa_\eta(\pi) \) are large.

In the original code DOPRI5 an approximation of the absolute value of the dominant eigenvalue is computed by using the already computed quantities \( k_i, k_{i0}, g_7, g_6 \):

\[
|\lambda| = \frac{|k_7 - k_6|}{||g_7 - g_6||}. \tag{14}
\]

Once \( |\lambda| \) is computed the product \( h|\lambda| \) is compared to the boundary of stability domain of the method in order to detect if the stepszie is limited by stability reason; the practical test is: \( h|\lambda| > 3.25 \) for at least 15 steps. These steps could be consecutive or separated by at most 5 steps where the inequality is not satisfied. In the modified version we compute two approximations of the dominant eigenvalue, \( |\lambda_y| \) and \( |\lambda_y| \), using in the appropriate way formula \((14)\), and these approximations can be used as in the standard stiffness detection algorithm implemented in DOPRI5.

So to give to the user all the important information about the solution, we have included the following tests:

\[
\begin{aligned}
S1: \text{The stepsize is limited by stability reason if } e_y(h) < 0.1e_z(h); & \text{ for at least 50 steps. These steps could be consecutive or separated by at most 5 steps where the inequality is not satisfied.} \\
S2: \text{The stepsize is limited by stability reason if } 2.8 < h \max(|\lambda_y|, |\lambda_y|) < 4.2 \text{, for at least 25 steps. These steps could be consecutive or separated by at most 5 steps where the inequality is not satisfied.} \\
S3: \text{The numerical solution is unstable if } r_z(\pi) > 10^{10} \text{ and } \kappa_\eta(\pi) > 10^8.
\end{aligned}
\]
Concerning the computational cost, at each step we doubled the number of function evaluations. This is a very modest price to pay for explaining the behavior of the numerical solution and giving information about the conditioning and the stiffness of the problem, keeping in mind that the direct use of an implicit method is much more expensive.

3.1. Numerical experiments

In the following we present some numerical results to demonstrate the performance of the new mesh selection algorithm and of the stiffness detection algorithm based on the computation of \( \sigma(T) \), the computation of an approximation of \( \sigma_c(t_0, t_f) \) will be described in the next section. Comparisons are made between the original code and the modified one. For simplicity, in the following we omit the dependence on \( \pi \) for the stiffness ratio and the conditioning parameters.

The results are tabulated in the tables for different values of \( rtol \) and \( atol \). In the tables we show the time \( t \) of the integration when stiffness is detected or when the code recognize that the stepsize is restricted only by stability reason. We denote it by \( t_i \) for the original code and \( t_m \) for the new one (\(-\) in the tables means that the code did not detect stiffness and \(+\) means that the code failed to compute the solution), the \((e)\), \((i)\) or \((\sigma)\) after \( t_m \) denote how the stopping decision has been made, \((e)\) if \( S1 \) is satisfied, \((i)\) if \( S2 \) is satisfied, \((\sigma)\) if \( \sigma_o \) computed at \( t_m \) is greater than 50 and \( r_t < 10^{-13} \). \( N_o, N_m \) are the number of total steps required by the original and the modified code. We also compute the errors \( E_o \) and \( E_m \) for the original and the modified code by using the following formula:

\[
\max_{0 \leq j \leq n} \left( \frac{|y(t_j) - y_h(t_j)|}{(atol/rtol + |y(t_j)|)} \right),
\]

where \( t_j, i = 1, \ldots, n \) are the mesh points in the final mesh, \( y_h \) represented the numerical solution and \( y \) is an accurate solution computed by an implicit solver such as \texttt{ode15s} (MATLAB solver for solving stiff problems) with very small tolerance \( rtol = atol = 10^{-13} \).

We run the original code \texttt{DOPRI5} using the parameter Nonstiff = 10, in order to start the stiffness detection at the beginning of the integration.

**Example 1.** Classical problem due to Robertson [6] which models a chemical reaction. The equations and initial values are given by

\[
\begin{align*}
y'_1 &= -0.04y_1 + 10^5y_2y_3, \\
y'_2 &= 0.04y_1 - 10^5y_2y_3 - 3 \cdot 10^7y_2^2, \quad t \in [0, 10] \\
y'_3 &= 3 \cdot 10^7y_2^2, \\
y(0) &= (1, 0, 0)^T.
\end{align*}
\]

**Example 2.** The Brusselator problem modeled as (for more details see page 6 in [6])

\[
\begin{align*}
\frac{\partial u}{\partial t} &= A + u^2v - (B + 1)u + \varepsilon \frac{\partial^2 u}{\partial x^2}, \quad t \in [0, 10] \\
\frac{\partial v}{\partial t} &= Bu - u^2v + \varepsilon \frac{\partial^2 v}{\partial x^2},
\end{align*}
\]

where \( u \) and \( v \) denote the concentration of the reaction products, \( A \) and \( B \) denote the concentration of input reagents. In our work we choose \( A = 1, B = 3 \) and \( \varepsilon = 0.02 \). The initial conditions are \( u(x, 0) = 1 + \sin(2\pi x), v(x, 0) = 3 \).

**Example 3.** The stiff Beam which is originally described by a partial differential equation subject to boundary conditions. It describes the motion of an elastic, inextensible thin beam clamped at one end and subject to a force acting at the free end. The semi-discretization in space of this equation leads to a stiff system of 80 nonlinear differential equations. A complete description of the problem could be found in [25] and [6, p.8].

**Example 4.** The Model of Flame Propagation given by [10,31]

\[
\begin{align*}
y' &= y^2 - y^3, \quad t \in [0,2/\delta] \\
y(0) &= \delta.
\end{align*}
\]

By looking at Table 1 we see that for the Robertson problem for \( rtol = atol = 1e-4 \), the original code fails to give a solution, but the new code, which implements a different mesh selection strategy, is able to give a solution, which is more accurate than was required, and this is what we expect, since the stepsize is limited by stability reason. Moreover, we observe that the value of \( \sigma_o(t_m) \) (that is \( \sigma_o \) computed in the interval \([t_0, t_m]\)) is greater than \( \sigma_o(t_f) \), this means that \( \gamma_y \) is growing. We expect that if we integrate in a bigger interval an instability will be detected. For smaller values of the tolerances the conditioning parameters stabilized and this give an indication about the reliability of the solution.

We obtain similar results for the Brusselator problem, but looking at Fig. 1, we see that, using $atol = rtol = 8 \cdot 10^{-2}$, the new algorithm provides a more accurate solution without oscillation, with a smaller number of mesh points and a relative error $E_m = 3.27 \cdot 10^{-7}$, the original code instead compute a solution with a relative error $E_m = 6.89 \cdot 10^{-1}$. This means that the new mesh selection strategy based on conditioning gives a more accurate solution using the same number of mesh points. When $atol = rtol = 10^{-8}$ the stepsize is not restricted by stability reason, but the degree of stiffness of the problem does not change, with respect to the one computed using higher tolerances. In this case $\sigma_q \approx 20$, so we can consider this problem moderately stiff. Moreover, for $N = 80$, the stiffness ratio increases and the new algorithm computes a solution which is much more accurate than required. In this case the new step selection always generate a solution with an error of the order of $10^{-8}$, the original step selection algorithm instead generated, using a similar number of mesh points, a solution with an error which is much higher.

A similar behavior in the error is observed for the Beam problem, were the error computed with the original algorithm is $10^4$ times higher than the required input tolerance, the new mesh selection algorithm instead compute the solution with an error which is usually smaller than the required tolerances with the same number of mesh points. For the beam problem, we also see that the original code is not able to detect if the stepsize is limited by stability reason, the new one, instead, detect soon that the stepsize is limited by stability reason, the value of $\sigma_q$ is however small. We observe that the algorithm compute a lower bound of $\sigma_q(T)$.

For the flame propagation problem, the stiffness depends on the value of $\delta$ (Table 2). For $\delta = 0.1$ the problem is not stiff, the value of $\kappa_q \gg 1$ means that the problem has a growing solution in the interval. If we decrease $\delta$ the code not only detect stiffness, but also the ill-conditioning of the problem, in fact both $\kappa_q$ and $\sigma_q$ grow and $\kappa_q \approx 1.46 \cdot 10^2$ for $\delta = 10^{-4}$. This problem has been carefully described in [10,31] as an illustrative example of stiffness. In Fig. 2 we report the numerical solution computed with $\delta = 10^{-4}$, and $rtol = atol = 10^{-4}$ by the two codes and a zoom of the solution near $t = \delta$. We note that the modified code computes a more accurate solution with the same number of mesh points.

### 4. Generalization for non-uniformly stable problems

The following two examples show that the algorithm presented in the previous section works very well for uniformly asymptotically stable problems but could give a value of $\sigma_q$ smaller than we could aspect for stable, but not uniformly stable, problems.

---

**Table 1**

<table>
<thead>
<tr>
<th>$t_f$</th>
<th>$n_m$</th>
<th>$rtol$</th>
<th>$atol$</th>
<th>$\sigma_q(t_m)$</th>
<th>$\sigma_q$</th>
<th>$\kappa_q$</th>
<th>$\gamma_q$</th>
<th>$N$</th>
<th>$N_m$</th>
<th>$E_0$</th>
<th>$E_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robertson, $t_f = 10$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$- (\epsilon)$</td>
<td>0.04311 0.04780</td>
<td>1e-4 1e-7</td>
<td>1.00e2</td>
<td>1.00e0</td>
<td>1.20e-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05009</td>
<td>0.04799</td>
<td>1e-5 1e-8</td>
<td>5.92e1</td>
<td>1.12e4</td>
<td>1.00e0</td>
<td>8.92e-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.03484</td>
<td>0.04924</td>
<td>1e-6 1e-9</td>
<td>5.71e1</td>
<td>9.98e3</td>
<td>1.00e0</td>
<td>1.00e-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brusselator, $N = 40$, $t_f = 10$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.69436</td>
<td>0.75636</td>
<td>8e-2 8e-2</td>
<td>6.84e0</td>
<td>1.81e1</td>
<td>1.00e0</td>
<td>5.52e-2</td>
<td>498</td>
<td>412</td>
<td>6.86e-1</td>
<td>3.27e-7</td>
<td></td>
</tr>
<tr>
<td>0.60002</td>
<td>0.90407</td>
<td>1e-4 1e-7</td>
<td>6.73e0</td>
<td>1.93e1</td>
<td>1.00e0</td>
<td>5.17e-2</td>
<td>415</td>
<td>424</td>
<td>2.84e-4</td>
<td>9.14e-7</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>1e-8 1e-8</td>
<td>1.93e1</td>
<td>1.00e0</td>
<td>5.17e-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brusselator, $N = 80$, $t_f = 10$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.21481</td>
<td>0.18269</td>
<td>8e-2 8e-2</td>
<td>6.24e0</td>
<td>4.72e1</td>
<td>1.00e0</td>
<td>2.12e-2</td>
<td>1791</td>
<td>1593</td>
<td>8.37e-1</td>
<td>2.15e-8</td>
<td></td>
</tr>
<tr>
<td>0.15044</td>
<td>0.23545</td>
<td>1e-4 1e-7</td>
<td>7.02e0</td>
<td>4.85e1</td>
<td>1.00e0</td>
<td>2.06e-2</td>
<td>1593</td>
<td>1603</td>
<td>4.39e-4</td>
<td>1.10e-9</td>
<td></td>
</tr>
<tr>
<td>1.48430</td>
<td>0.35983</td>
<td>1e-8 1e-8</td>
<td>9.05e0</td>
<td>4.83e1</td>
<td>1.00e0</td>
<td>2.07e-2</td>
<td>1599</td>
<td>1602</td>
<td>3.05e-8</td>
<td>1.65e-8</td>
<td></td>
</tr>
<tr>
<td>Beam, $t_f = 0.5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>0.01574</td>
<td>1e-4 1e-4</td>
<td>1.67e0</td>
<td>1.99e0</td>
<td>2.72e1</td>
<td>1.36e1</td>
<td>2491</td>
<td>2491</td>
<td>1.19e-1</td>
<td>2.66e-7</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>0.01514</td>
<td>1e-6 1e-6</td>
<td>1.62e0</td>
<td>1.98e0</td>
<td>2.65e1</td>
<td>1.34e1</td>
<td>2458</td>
<td>2458</td>
<td>1.18e-3</td>
<td>2.60e-9</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>0.01484</td>
<td>1e-8 1e-8</td>
<td>1.64e0</td>
<td>1.98e0</td>
<td>2.65e1</td>
<td>1.34e1</td>
<td>2469</td>
<td>2460</td>
<td>1.15e-5</td>
<td>2.60e-7</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2**

<table>
<thead>
<tr>
<th>$\Delta$</th>
<th>$t_m$</th>
<th>$t_n$</th>
<th>$\sigma_q(t_m)$</th>
<th>$\sigma_q$</th>
<th>$\kappa_q$</th>
<th>$\gamma_q$</th>
<th>$N$</th>
<th>$N_m$</th>
<th>$E_0$</th>
<th>$E_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flame propagation, $rtol = 1e-4$, $atol = 1e-7$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e-1</td>
<td>-</td>
<td>-</td>
<td>3.24e0 1.62e1</td>
<td>5.00e0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e-2</td>
<td></td>
<td></td>
<td>2.86e1 1.43e3</td>
<td>5.02e1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e-3</td>
<td>1047.192</td>
<td>1042.907</td>
<td>1.51e2 2.95e2</td>
<td>1.48e5 5.02e2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e-4</td>
<td>1008.990</td>
<td>1002.069</td>
<td>1.47e3 2.93e3</td>
<td>1.46e7 4.99e3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 5. The Kreiss problem [6] which is a linear and non-autonomous problem:
\[ y' = A(t)y, \quad t \in [0, 4\pi], \quad y(0) = (0, 1)^T, \tag{18} \]
where
\[ A(t) = Q^T(t) \Lambda \epsilon Q(t), \]
and
\[ Q(t) = \begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}, \quad \Lambda \epsilon = \begin{pmatrix} -1 & \epsilon^{-1} \\ -\epsilon^{-1} & -1 \end{pmatrix}. \]
with \( \epsilon = 10^{-3} \).

Example 6. A non-uniformly stable test problem:
\[ y' = e^t \cos y, \quad t \in [0, t_f] \]
\[ y(0) = 0. \tag{19} \]

From Tables 3 and 4 we see that for the uniformly stable Kreiss problem, the value of \( \sigma_{\epsilon} \) decreases with the width of the interval and remains the same if we change \( t_0 \) and \( t_f \), but we leave unchanged the width \( t_f - t_0 \). Different results are obtained for the non-uniformly stable problem (see Tables 5 and 6). In this case changing the value of \( t_0 \), the value of \( \sigma_{\epsilon} \) increases even if the width of the interval decreases. Example 6 shows that Definition 1 does not give the complete information about
stiffness if the problem is not uniformly stable. For this reason we have introduced the new general Definition 2. For uniformly stable problems the approximation of \( \sigma_i(T) \) could be efficiently done using the algorithm presented in Section 3. This is not the case for general stiff problems where it is important to compute \( k_r(t, t_f, \eta) \) and \( \gamma_r(t, t_f, \eta) \) changing the value for \( t \geq t_0 \) and this is not computationally efficient. So, we decide to dynamically choose a discrete set of times, say \( N_\epsilon \), and to compute an approximation of \( \sigma_r(T, t_f) \)

\[
\sigma_r(T) = \max_{t_0 \leq t \\ t \in \mathbb{N}_e \cap (\eta(t_f) - \eta(t_0)) \left( \frac{k_r(t_0, t_f)}{\eta(t_f) - \eta(t_0)} \right)
\]

where \( k_r \) is the mesh used in the interval \( t_r, t_{r+1} \). The new algorithm starts with \( t_r = t_0, i = 1 \) and chooses the new point \( t_{r+1} \) by computing \( r_i \), the relative difference between the numerical approximation of \( y \) and \( y(t) \), at time \( t_r \), if this difference becomes negligible, no more information is added to the computation of \( \sigma_r(T) \) and, if \( \sigma_r(T) \) is small we restart the computation. The results of this algorithm for problem (6) are reported in Table 7. We see that now the value of \( \sigma_r(T) \) changes and the stiffness is detected. For all the problems presented in the previous section the results are the same, because the restart is not needed. We observe that the stiffness strongly depends on the size of the interval when the solution as to be computed, in Table 8 we report the results of the Robertson problem using a different value of \( t_f \). When \( t_f \) is less than \( 10^{-4} \) the problem is
not stiff, the value of $\sigma^G$ is small and the explicit method works well, in the other cases $\sigma^G$ grows and the numerical method require an higher computational cost. The new mesh selection algorithm gives a solution whose error does not depends on the input tolerances, because the stepsize is always restricted by stability reason. For higher values of the tolerance the original code gives a solution with an higher error, using a similar number of mesh points.

5. Conclusions

We have presented a new mesh selection algorithm based on conditioning that gives information about the conditioning and the stiffness of the problem. This algorithm has been implemented into the Matlab code DOPR5 and the R code cashkarp.

We observe that, as pointed out in [23] “when an initial value problem is run on a computer, the results may appear plausible even if they are unreliable because of some unrecognized numerical instability”. Additional information about the behavior of the numerical solution are, therefore, of great interest for potential users of a numerical computer code. The presented algorithm not only computes the conditioning parameters, but also gives information about the stiffness and disclose unstable situations, providing a numerical solution which is, in general, much more accurate than the one computed by standard mesh selection algorithms, that do not take into account the conditioning.

Acknowledgment

The authors are very grateful to Prof. L. Brugnano for his comments and suggestions during the preparation of this paper. This work is devoted to the memory of Prof. Donato Trigiante, remembering the scientific discussions with him.

References


