Applied Mathematics and Computation 255 (2015) 125–134

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

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

ARTICLE INFO

Keywords:
Mesh selection
Conditioning
Stiffness
ODE problems
Initial value problems
Runge–Kutta methods

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

$$y' = f(t,y),\ \ y(t_0) = y_0, \quad t \in [t_0, t_f], \tag{1}$$

where y_0 is a given vector in 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 $err = O(h^p)$, $\widetilde{err} = O(h^q)$, with $q < p$, usually $err < \widetilde{err}$, if the stepsize is limited by stability requirements and $err \gg \widetilde{err}$ 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

* Corresponding author.
E-mail addresses: francesca.mazzia@uniba.it (F. Mazzia), abdelhameed_nagy@yahoo.com (A.M. Nagy).

http://dx.doi.org/10.1016/j.amc.2014.03.065
0096-3003/

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). 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.

Keywords: mesh selection, conditioning, stiffness, ODE problems, initial value problems, Runge-Kutta methods

1. Introduction

We consider initial value problems for systems of ordinary differential equations

$$y' = f(t, y), \qquad y(t_0) = y_0, \qquad t \in [t_0, t_f], \tag{1}$$

where y_0 is a given vector in 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, 2, 3, 4, 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 $err = O(h^p), \widetilde{err} = O(h^q)$, with $q < p$, usually $err < \widetilde{err}$, if the stepsize is limited by stability requirements and $err \gg \widetilde{err}$ 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 [8, 9, 2]. 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

*Corresponding authors
Email addresses: mazzia@dm.uniba.it (Francesca Mazzia), abdelhameed_nagy@yahoo.com (A. M. Nagy)

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 the local error. The definition of the conditioning parameters has been given [15, 16, 14] and they measure the sensitivity of the problem to perturbations. Such parameters can be defined both for the continuous 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 `bvptwp` [17] and `tom` [18], the R code `bvptwp` of the R package **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 beetwen 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 stiffness for uniformly asymptotically stable problem. 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:

$$y' = A(t)y + q(t), \qquad y(t_0) \quad = \quad y_0, \qquad t \in [t_0, t_f] \tag{2}$$

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

$$\kappa_c = \sup_{t_0 \le x \le t < \infty} \|Y(t)Y^{-1}(x)\|,$$

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 [12, p.239], [11] and [26, p.100] for more details). Using κ_c it is possible to give a bound on the effect of perturbation η on the initial condition y_0. In fact, if we consider the perturbed problem:

$$\hat{y}' = A(t)\hat{y} + q(t), \qquad \hat{y}(t_0) = y_0 + \eta, \qquad t \in [t_0, t_f], \tag{3}$$

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

$$z' = A(t)z, \qquad z(t_0) = \eta, \qquad t \in [t_0, t_f], \tag{4}$$

and

$$\|z\|_\infty \le \kappa_c \|\eta\|.$$

For uniformly asymptotically stable problems we have that there exist $\alpha > 0$, $\beta > 0$ such that $\|Y(t)Y^{-1}(x)\| \leq \beta e^{-\alpha(t-x)}$ for all $x \geq t$, this means that $\kappa_c \leq \beta$ and we say that the IVP is well-conditioned if β 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., [14, 13]):

$$\begin{aligned} \kappa_c(t_0, t_f, \eta) &= \frac{1}{\|\eta\|} \max_{t_0 \leq t \leq t_f} \|z(t)\|, & \bar{\kappa}_c(t_0, t_f) &= \max_{\|\eta\| \leq \delta, \eta \neq 0} \kappa_c(t_0, t_f, \eta), \\ \gamma_c(t_0, t_f, \eta) &= \frac{1}{(t_f - t_0)\|\eta\|} \int_{t_0}^{t_f} \|z(t)\| dt, & \bar{\gamma}_c(t_0, t_f) &= \max_{\|\eta\| \leq \delta, \eta \neq 0} \gamma_c(t_0, t_f, \eta), \end{aligned} \tag{5}$$

where $\|\cdot\|$ is a compatible vector norm and δ 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:*

$$\sigma_c(T) = \max_{\|\eta\| \leq \delta, \eta \neq 0} \frac{\kappa_c(t_0, t_0 + T, \eta)}{\gamma_c(t_0, t_0 + T, \eta)} \gg 1. \tag{6}$$

We note that $\sigma_c(T)$ depends on the width of the interval of integration. More details about $\sigma_c(T)$ 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*

$$\sigma_c^G(t_0, t_f) = \max_{t_0 \leq t < t_f} \max_{\|\eta\| \leq \delta, \eta \neq 0} \frac{\kappa_c(t, t_f, \eta)}{\gamma_c(t, t_f, \eta)} \gg 1. \tag{7}$$

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:

$$\hat{y}' = \quad f(t, \hat{y}), \qquad \hat{y}(t_0) = y_0 + \eta, \qquad t \in [t_0, t_f] \tag{8}$$

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

$$z' = f(t, \hat{y}) - f(t, y), \qquad z(t_0) = \eta, \qquad t \in [t_0, t_f] \tag{9}$$

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, 28, 29, 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), (6) and (7). To do this we also compute an approximation of $\hat{y}$, the solution of (8), by choosing a suitable value of the perturbation η. To compute an approximation of $\hat{y}$ that satify the same input error tolerance we estimate the error for $\hat{y}$ and we choose the stepsize accordly. Moreover, the estimation of the conditioning parameter is based on the difference $z = \hat{y} - y$ computed using equation (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,

$$e_y(h) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \Big(\frac{E_{n,i}(h)}{\rho_i} \Big)^2}, \tag{10}$$

where $\rho_i = atol_i + max(|y_{n,i}|, |y_{n+1,i}|) rtol_i$ and $E_{n,i}(h)$ 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 is necessary to use a smaller absolute tolerance, since z is the difference beetwen two solutions that are very close each other, so the error estimation for z is computed using the absolute tolerance equal to $10^{-2} atol_i$. This generate a new mesh selection strategy:

$$h_{new} = fac \cdot h \cdot e(h)^{-1/5}, \quad e(h) = \max(e_y(h), e_{\hat{y}}(h), e_z(h)).$$

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:

$$\kappa_\eta(\pi) = \frac{1}{\|\eta\|_2} \max_{i=1,\ldots,N} (\|z_i\|_2), \tag{11}$$

$$\gamma_\eta(\pi) = \frac{1}{\|\eta\|_2} \frac{1}{(t_N - t_0)} \sum_{i=1}^{N} \frac{h_i}{2} (\|z_i\|_2 + \|z_{i-1}\|_2), \tag{12}$$

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:

$$r_z(\pi) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \Big(\frac{y_{n,i} - \hat{y}_{n,i}}{10^{-2} atol_i + |y_{n,i}| rtol_i} \Big)^2}.$$

In practice we compute two solutions of the same problem starting with a different initial condition. It is important to detect the choice of η that yields a good approximation of $\sigma_c(T)$. After some extensive numerical experiments we choose η 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 ξ we require $\xi > 10^4 \cdot eps$, where 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 stepsize, the chosen perturbation η 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 approximation of the erros $e_y(h)$ 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_7, k_6, g_7, g_6:

$$|\tilde{\lambda}| = \frac{\|k_7 - k_6\|}{\|g_7 - g_6\|}. \tag{14}$$

Once $|\tilde{\lambda}|$ is computed the product $h|\tilde{\lambda}|$ is compared to the boundary of stability domain of the method in order to detect if the stepsize is limited by stability reason; the practical test is: $h|\tilde{\lambda}| > 3.25$ 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, $|\tilde{\lambda}_y|$ and $|\tilde{\lambda}_{\hat{y}}|$, 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:

S1: The stepsize is limited by stability reason if $e_y(h) < 0.1e_z(h)$; for at least 50 steps. These steps could be consecutive or separated by at most 5 steps where the inequality is not satisfied;

S2: The stepsize is limited by stability reason if $2.8 < h\max(|\tilde{\lambda}_y|, |\tilde{\lambda}_{\hat{y}}|) < 4.2$, for at least 25 steps. These steps could be consecutive or separated by at most 5 steps where the inequality is not satisfied.

S3: The numerical solution is unstable if $r_z(\pi) > 10^{10}$ and $\kappa_\eta(\pi) > 10^8$;

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 $\sigma_c(T)$, the computation of an approximation of $\sigma_c^G(t_0, t_f)$ 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 π 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_o 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), (λ) or (σ) after t_m denote how the stopping decision has been made, (e) if S1 is satisfied, (λ) if S2 is satisfied, (σ) if σ_η computed at t_m is greater than 50 and $r_z < 10^{-5}$; 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:

$$\max_{0\le i\le n}\left(\frac{|y(t_i)-y_\pi(t_i)|}{(atol/rtol+|y(t_i)|)}\right),$$

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

We run the original code `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*

$$\begin{aligned} &y_1' = -0.04y_1 + 10^4 y_2 y_3,\\ &y_2' = 0.04y_1 + 10^4 y_2 y_3 - 3\cdot 10^7 y_2^2, \qquad t\in[0,10]\\ &y_3' = 3\cdot 10^7 y_2^2,\\ &y(0) = (1,0,0)^T. \end{aligned} \tag{15}$$

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

$$\begin{aligned} &\frac{\partial u}{\partial t} = A + u^2 v - (B+1)u + \alpha\frac{\partial^2 u}{\partial x^2}, \qquad t\in[0,10]\\ &\frac{\partial v}{\partial t} = Bu - u^2 v + \alpha\frac{\partial^2 v}{\partial x^2}, \end{aligned} \tag{16}$$

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 $\alpha = 0.02$. 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]*

$$\begin{aligned} &y' = y^2 - y^3, \quad t\in[0,2/\delta]\\ &y(0) = \delta. \end{aligned} \tag{17}$$

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 $\sigma_\eta(t_m)$ (that is σ_η computed in the interval $[t_0, t_m]$) is greater than $\sigma_\eta(t_f)$, this means that γ_η is growing. We expect that if we integrate in a bigger interval an unstability will be detected. For smaller values of the

Table 1: Examples 1, 2, 3.

Robertson, $t_f = 10$											
t_o	t_m	$rtol$	$atol$	$\sigma_\eta(t_m)$	σ_η	κ_η	γ_η	N_o	N_m	E_o	E_m
– (*)	0.04311 (λ, σ)	1e-4	1e-4	6.56e1	8.31e0	1.0e0	1.20e-1	–	8992	–	2.27e-5
0.02661	0.04780 (λ, σ)	1e-4	1e-7	1.06e2	1.05e4	1.0e0	9.53e-5	7156	7103	8.24e-5	8.01e-7
0.03509	0.04799 (λ, σ)	1e-5	1e-8	5.92e1	1.12e4	1.0e0	8.92e-5	7103	7105	9.91e-6	5.27e-7
0.03844	0.04924 (λ, σ)	1e-6	1e-9	5.71e1	9.98e3	1.0e0	1.00e-4	7102	7105	1.23e-6	5.82e-7
Brusselator, $N = 40$, $t_f = 10$											
0.69436	0.75636 (λ)	8e-2	8e-2	6.84e0	1.81e1	1.0e0	5.52e-2	498	412	6.86e-1	3.27e-7
0.60002	0.90407 (e)	1e-4	1e-7	6.73e0	1.93e1	1.0e0	5.17e-2	415	424	2.84e-4	9.14e-7
–	–	1e-8	1e-8	–	1.93e1	1.0e0	5.17e-2	421	421	1.65e-8	1.65e-8
Brusselator, $N = 80$, $t_f = 10$											
0.21481	0.18269 (λ)	8e-2	8e-2	6.24e0	4.72e1	1.0e0	2.12e-2	1791	1593	8.37e-1	2.15e-8
0.15044	0.23545 (e)	1e-4	1e-7	7.02e0	4.85e1	1.0e0	2.06e-2	1593	1603	4.39e-4	1.10e-9
1.48430	0.35983 (e)	1e-8	1e-8	9.05e0	4.83e1	1.0e0	2.07e-2	1599	1602	3.05e-8	2.91e-9
Beam, $t_f = 0.5$											
–	0.01574 (e)	1e-4	1e-4	1.67e0	1.99e0	2.72e1	1.36e1	2451	2451	1.19e-1	2.66e-7
–	0.01514 (e)	1e-6	1e-6	1.62e0	1.98e0	2.65e1	1.34e1	2458	2458	1.18e-3	2.60e-7
–	0.01484 (e)	1e-8	1e-8	1.64e0	1.98e0	2.65e1	1.34e1	2469	2460	1.15e-5	2.60e-7

Table 2: Example 4.

Flame propagation, $rtol$ = 1e-4, $atol$ = 1e-7.										
δ	t_o	t_m	$\sigma_\eta(t_m)$	σ_η	κ_η	γ_η	N_o	N_m	E_o	E_m
1e-1	–	–	–	3.24e0	1.62e1	5.00e0	24	29	6.37e-5	1.39e-5
1e-2	192.725	–	–	2.86e1	1.43e3	5.02e1	64	82	1.20e-4	1.07e-4
1e-3	1074.192	1024.907 (σ)	1.51e2	2.95e2	1.48e5	5.02e2	347	371	1.72e-3	1.61e-3
1e-4	10084.990	10023.069 (σ)	1.47e3	2.93e3	1.46e7	4.99e3	3080	3111	4.94e-2	4.81e-2

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 Figure 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 $E_o = 6.89 \cdot 10^{-1}$. 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 no 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 $\sigma_\eta \approx 20$, 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 10^3 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 σ_η is however small. We observe that the algorithm compute a lower bound of $\sigma_c(T)$.

For the flame propagation problem, the stiffness depends on the value of δ. For $\delta = 0.1$ the problem is not stiff, the value of $\kappa_\eta \gg 1$ means that the problem has a growing solution in the interval. If we decrease δ the code not only detect stiffness, but also the ill-conditioning of the problem, in fact both κ_η and σ_η grow and $\kappa_\eta = 1.46 \cdot 10^7$ for $\delta = 10^{-4}$. This problem has been carfully described in [31, 10] as an illustrative example of stiffness. In Figure 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.

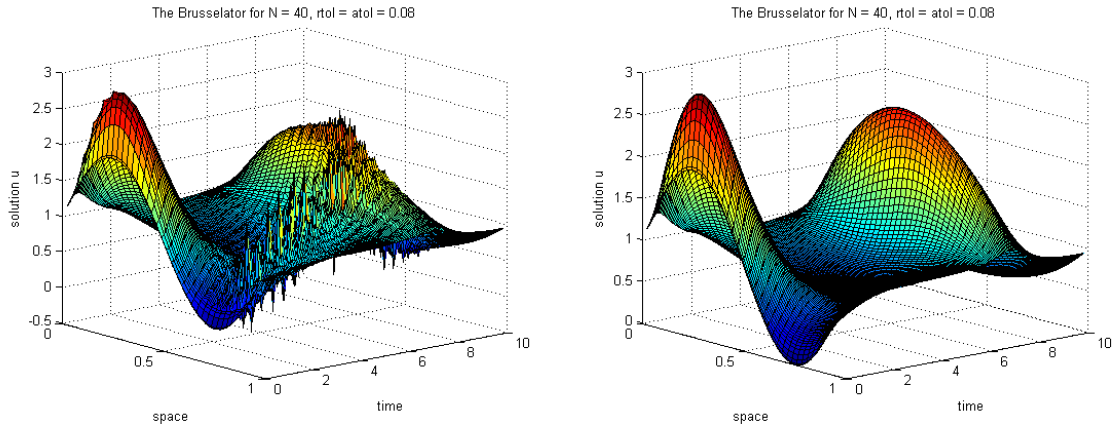

Figure 1: The numerical solution for equation (16) using the original (left) and the modified DOPRI5 code (right).

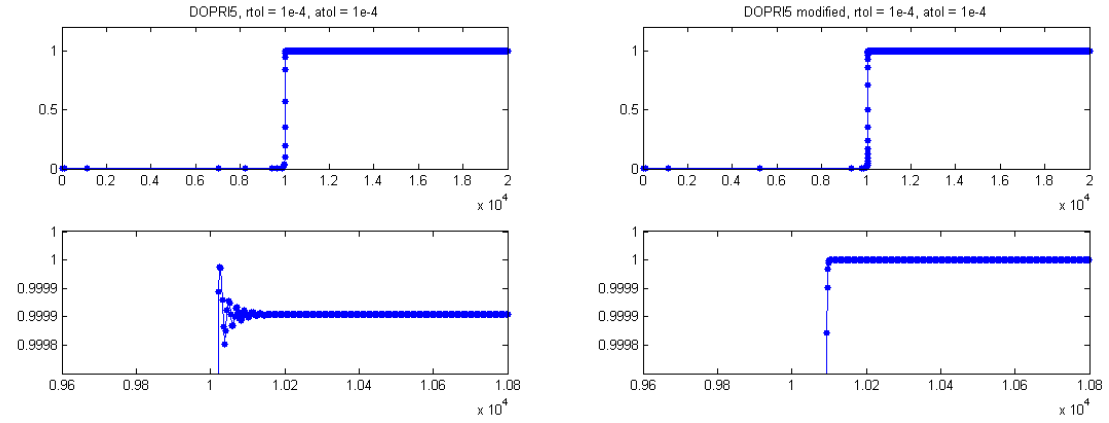

Figure 2: The numerical solution for Example 4 using the original (left) and the modified DOPRI5 code (right). The figures at the bottom are a zoom of the solution, the width of the y-axis is $(1 - 2 \cdot 10^{-4}, 1 + 5 \cdot 10^{-5})$.

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 σ_η smaller than we could aspect for stable, but not uniformly stable, problems.

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

$$Q(t) = \begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix}, \quad \Lambda_\epsilon = \begin{pmatrix} -1 & \\ & -\epsilon^{-1} \end{pmatrix},$$

with $\epsilon = 10^{-3}$.

Example 6. *A non-uniformly stable test problem:*

$$\begin{aligned} &y' = e^t \cos y, \quad t \in [0, t_f] \\ &y(0) = 0. \end{aligned} \tag{19}$$

Table 3: Kreiss problem, $t_0 = 0$ and $t_f = 10$.

$rtol$	$atol$	t_o	t_m	$\sigma_\eta(t_m)$	σ_η	κ_η	γ_η	N_o	N_m	E_o	E_m
1.0e-3	1.0e-5	0.0913	0.1045 (λ, σ)	9.46e1	5.35e3	1.0e0	1.87e-4	3038	3038	1.45e-3	1.45e-3
1.0e-4	1.0e-6	0.0817	0.1082 (λ, σ)	9.90e1	5.48e3	1.0e0	1.82e-4	3045	3045	9.95e-5	7.89e-5
1.0e-5	1.0e-7	0.0753	0.1118 (λ, σ)	1.02e2	4.37e3	1.0e0	2.29e-4	3054	3054	9.14e-6	7.28e-6
1.0e-6	1.0e-8	1.1070	0.1057 (λ, σ)	9.80e1	5.53e3	1.0e0	1.81e-4	3085	3085	8.21e-7	8.21e-7
1.0e-7	1.0e-9	3.4090	0.6644 (λ, σ)	4.78e2	5.53e3	1.0e0	1.81e-4	3200	3200	8.69e-8	8.69e-8

From Tables 3-4 we see that for the uniformly stable Kreiss problem, the value of σ_η 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 5-6). In this case changing the value of t_0, the value of σ_η 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 $\sigma_c(T)$ 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 $\kappa_c(t, t_f, \eta)$ and $\gamma_c(t, t_f, \eta)$ 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_σ, and to compute an approximation of $\sigma_c^G(t_0, t_f)$

$$\sigma_\eta^G(\pi_i) = \max_{1 \leq i \leq N_\sigma} \frac{\kappa_\eta(\pi_i)}{\gamma_\eta(\pi_i)},$$

where π_i is the mesh used in the interval $t_i^\sigma, t_{i+1}^\sigma$, The new algorithm starts with $t_1^\sigma = t_0$, $i = 1$ and chooses the new point t_{i+1}^σ by computing r_z, the relative difference between the numerical approximation of y and $\hat{y}$, at time t_{i+1}^σ, if this difference became negligible, no more information is added to the computation of $\sigma_\eta^G(\pi_i)$ and, if $\sigma_\eta^G(\pi_i)$ 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 σ_η^G 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^{-1} the problem is not stiff, the value of σ_c^G is small and the explicit method works well, in the other cases σ_c^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 DOPRI5 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

Table 4: Kreiss problem, $rtol$ = 1e-4 and $atol$ = 1e-8.

t_0	t_f	t_o	t_m	$\sigma_\eta(t_m)$	σ_η	κ_η	γ_η	N_o	N_m	E_o	E_m
0.0e0	1.0e1	0.2044	0.1069 (λ, σ)	9.83e1	5.50e3	1.0e0	1.82e-4	3060	3060	1.86e-4	1.86e-4
2.0e0	1.0e1	2.0696	2.1073 (λ, σ)	9.85e1	4.40e3	1.0e0	2.27e-4	2488	2496	4.26e-4	2.18e-4
4.0e0	1.0e1	4.0742	4.1065 (λ, σ)	9.79e1	3.30e3	1.0e0	3.03e-4	1871	1885	5.32e-4	7.10e-4
6.0e0	1.0e1	6.6002	6.1028 (λ, σ)	9.47e1	2.22e3	1.0e0	4.51e-4	1263	1267	4.72e-4	2.99e-4
8.0e0	1.0e1	8.1792	8.1034 (λ, σ)	9.52e1	1.17e3	1.0e0	8.55e-4	635	642	3.48e-4	2.38e-4
0.0e0	1.0e1	0.2044	0.1069 (λ, σ)	9.83e1	5.50e3	1.0e0	1.82e-4	3060	3060	1.86e-4	1.86e-4
2.0e0	1.2e1	2.0696	2.1073 (λ, σ)	9.85e1	5.50e3	1.0e0	1.82e-4	3095	3103	4.26e-4	2.18e-4
4.0e0	1.4e1	4.0742	4.1065 (λ, σ)	9.79e1	5.50e3	1.0e0	1.82e-4	3089	3103	5.32e-4	7.10e-4
6.0e0	1.6e1	6.6002	6.1028 (λ, σ)	9.47e1	5.50e3	1.0e0	1.82e-4	3093	3097	4.72e-4	2.99e-4
8.0e0	1.8e1	8.1792	8.1034 (λ, σ)	9.52e1	5.50e3	1.0e0	1.82e-4	3094	3102	3.48e-4	2.38e-4

Table 5: Example 6, $t_0 = 0$ and $t_f = 10$.

$rtol$	$atol$	t_o	t_m	$\sigma_\eta(t_m)$	σ_η	κ_η	γ_η	N_o	N_m	E_o	E_m
1.0e-3	1.0e-5	4.1876	4.5197 (λ)	5.21e0	1.15e1	1.0e0	8.71e-2	6678	6679	1.11e-3	1.05e-3
1.0e-4	1.0e-6	4.4242	4.5621 (λ)	5.27e0	1.15e1	1.0e0	8.66e-2	6683	6683	1.06e-4	1.09e-4
1.0e-5	1.0e-7	4.2311	4.6220 (λ)	5.35e0	1.16e1	1.0e0	8.65e-2	6689	6689	1.13e-5	1.18e-5
1.0e-6	1.0e-8	4.2816	4.6479 (λ)	5.37e0	1.16e1	1.0e0	8.65e-2	6697	6699	1.15e-6	1.08e-6
1.0e-7	1.0e-9	4.3511	4.6987 (λ)	5.42e0	1.15e1	1.0e0	8.66e-2	6710	6709	9.97e-8	1.10e-7

Table 6: Example 6 changing t_0, $t_f = 10$, $rtol$ = 1e-4 and $atol$ = 1e-8.

t_0	t_o	t_m	$\sigma_\eta(t_m)$	σ_η	κ_η	γ_η	N_o	N_m	E_o	E_m
0.0e0	4.3833	4.5625 (λ)	5.27e0	1.15e1	1.0e0	8.66e-2	6683	6683	1.09e-4	1.15e-4
2.0e0	4.3301	4.6909 (λ)	1.43e1	4.25e1	1.0e0	2.35e-2	6677	6677	9.68e-5	1.05e-4
4.0e0	4.8909	5.0511 (λ)	3.68e1	2.07e2	1.0e0	4.82e-3	6661	6661	1.23e-4	1.22e-4
6.0e0	6.1686	6.2026 (σ)	5.15e1	9.48e2	1.0e0	1.05e-3	6555	6555	7.51e-5	6.21e-5
8.0e0	8.0276	8.0276 (σ)	5.18e1	2.93e3	1.0e0	3.41e-4	5774	5774	9.80e-5	7.50e-5

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.

Acknowledgements

The authors are very grateful to Prof. L. Brugnano for his comments and suggestions during the preparation of this paper.

References

[1] L. F. Shampine, Stiffness and nonstiff differential equation solvers. II. Detecting stiffness with Runge-Kutta methods, ACM Trans. Math. Software 3 (1) (1977) 44–53.

[2] P. Rentrop, Partitioned Runge-Kutta methods with stiffness detection and stepsize control, Numer. Math. 47 (4) (1985) 545–564. doi:10.1007/BF01389456.
URL http://dx.doi.org/10.1007/BF01389456

[3] L. F. Shampine, Diagnosing stiffness for Runge-Kutta methods, SIAM J. Sci. Statist. Comput. 12 (2) (1991) 260–272. doi:10.1137/0912015.
URL http://dx.doi.org/10.1137/0912015

[4] K. Ekeland, B. Owren, E. Øines, Stiffness detection and estimation of dominant spectra with explicit Runge-Kutta methods, ACM Trans. Math. Software 24 (4) (1998) 368–382. doi:10.1145/293686.287641.
URL http://dx.doi.org/10.1145/293686.287641

[5] M. Sofroniou, G. Spaletta, Construction of explicit Runge-Kutta pairs with stiffness detection, Math. Comput. Modelling 40 (11-12) (2004) 1157–1169. doi:10.1016/j.mcm.2005.01.010.
URL http://dx.doi.org/10.1016/j.mcm.2005.01.010

[6] E. Hairer, G. Wanner, Solving ordinary differential equations. II, Vol. 14 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2010, stiff and differential-algebraic problems, Second revised edition, paperback.

[7] M. Sofroniou, G. Spaletta, Extrapolation methods in Mathematica, JNAIAM J. Numer. Anal. Ind. Appl. Math. 3 (1-2) (2008) 105–121.

[8] L. Petzold, Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations, SIAM J. Sci. Statist. Comput. 4 (1) (1983) 137–148. doi:10.1137/0904010.
URL http://dx.doi.org/10.1137/0904010

Table 7: Example 6 using the restarting algorithm, $t_0 = 0$ and $t_f = 10$.

$rtol$	$atol$	t_1^σ	$\sigma_\eta^G(t_0, t_1^\sigma)$	$\sigma_\eta^G(t_1^\sigma, t_f)$	κ_η	γ_η	N_o	N_m	E_o	E_m
1.0e-3	1.0e-5	5.50e0	6.33e0	1.23e3	1.0e0	8.16e-4	6678	6691	1.11e-3	1.05e-3
1.0e-4	1.0e-6	5.18e0	5.99e0	9.20e2	1.0e0	1.09e-3	6683	6697	1.06e-4	1.09e-4
1.0e-5	1.0e-7	5.18e0	5.99e0	9.06e2	1.0e0	1.10e-3	6689	6703	1.13e-5	1.18e-5
1.0e-6	1.0e-8	5.16e0	5.96e0	8.89e2	1.0e0	1.12e-3	6697	6713	1.15e-6	1.08e-6
1.0e-7	1.0e-9	5.11e0	5.90e0	8.58e2	1.0e0	1.17e-3	6710	6723	9.97e-8	1.10e-7

Table 8: Robertson problem, `DOPRI5` results.

t_0	t_f	t_o	t_m	$\sigma_\eta^G(t_m)$	σ_η^G	κ_η	γ_η	N_o	N_m	E_o	E_m
$rtol$ = 1e-4 and $atol$ = 1e-7											
0.0e0	2.0e-3	–	–	–	4.56e0	1.0e0	2.19e-1	8	15	1.57e-5	2.58e-8
0.0e0	1.0e-2	–	–	–	2.24e1	1.0e0	4.45e-2	14	25	4.88e-5	7.62e-8
0.0e0	1.0e-1	0.0266	0.0478 (λ, σ)	1.06e2	2.19e2	1.0e0	4.56e-3	76	85	8.07e-5	8.01e-7
0.0e0	5.0e0	0.0266	0.0478 (λ, σ)	1.06e2	6.37e3	1.0e0	1.57e-4	3419	3398	8.24e-5	8.01e-7
0.0e0	1.0e1	0.0266	0.0478 (λ, σ)	1.06e2	1.05e4	1.0e0	9.53e-5	7156	7103	8.24e-5	8.01e-7
$rtol$ = 1e-6 and $atol$ = 1e-10											
0.0e0	2.0e-3	–	–	–	2.28e0	1.0e0	4.40e-1	17	18	6.34e-8	4.16e-8
0.0e0	1.0e-2	–	–	–	1.11e1	1.0e0	9.02e-2	29	34	1.28e-7	4.16e-8
0.0e0	1.0e-1	0.0335	0.0515 (λ, σ)	5.71e1	1.11e2	1.0e0	9.02e-3	90	95	1.03e-6	6.08e-8
0.0e0	5.0e0	0.0335	0.0515 (λ, σ)	5.71e1	5.29e3	1.0e0	1.89e-4	3403	3407	1.03e-6	8.34e-7
0.0e0	1.0e1	0.0335	0.0515 (λ, σ)	5.71e1	1.03e4	1.0e0	9.66e-5	7108	7113	1.03e-6	8.33e-7

[9] J. C. Butcher, Order, stepsize and stiffness switching, Computing 44 (3) (1990) 209–220. doi:10.1007/BF02262217.
URL `http://dx.doi.org/10.1007/BF02262217`

[10] L. F. Shampine, S. Thompson, Stiff systems, Scholarpedia 2 (3) (2007) 2855.

[11] L. Brugnano, D. Trigiante, On the characterization of stiffness for ODEs, Dynam. Contin. Discrete Impuls. Systems 2 (3) (1996) 317–335.

[12] L. Brugnano, D. Trigiante, Solving differential problems by multistep initial and boundary value methods, Vol. 6 of Stability and Control: Theory, Methods and Applications, Gordon and Breach Science Publishers, Amsterdam, 1998.

[13] F. Iavernaro, F. Mazzia, D. Trigiante, Stability and conditioning in numerical analysis, JNAIAM J. Numer. Anal. Ind. Appl. Math. 1 (1) (2006) 91–112.

[14] L. Brugnano, F. Mazzia, D. Trigiante, Fifty years of stiffness, in: Recent advances in computational and applied mathematics, Springer, Dordrecht, 2011, pp. 1–21.
URL `\url{http://dx.doi.org/10.1007/978-90-481-9981-5_1}`

[15] J. R. Cash, F. Mazzia, Conditioning and Hybrid Mesh Selection Algorithms For Two Point Boundary Value Problems, Scalable Computing: Practice and Experience 10 (4) (2009) 347–361.

[16] J. R. Cash, F. Mazzia, N. Sumarti, D. Trigiante, The role of conditioning in mesh selection algorithms for first order systems of linear two point boundary value problems, J. Comput. Appl. Math. 185 (2) (2006) 212–224. doi:10.1016/j.cam.2005.03.007.
URL `http://dx.doi.org/10.1016/j.cam.2005.03.007`

[17] J. R. Cash, D. Hollevoet, F. Mazzia, A. M. Nagy, Algorithm 927: the MATLAB code `bvptwp.m` for the numerical solution of two point boundary value problems, ACM Trans. Math. Software 39 (2) (2013) Art. 15, 12. doi:10.1145/2427023.2427032.
URL `http://dx.doi.org/10.1145/2427023.2427032`

[18] F. Mazzia, D. Trigiante, A hybrid mesh selection strategy based on conditioning for boundary value ODE problems, Numer. Algorithms 36 (2) (2004) 169–187. doi:10.1023/B:NUMA.0000033132.99233.c8.
URL `http://dx.doi.org/10.1023/B:NUMA.0000033132.99233.c8`

[19] K. Soetaert, J. R. Cash, F. Mazzia, **bvpSolve**: Solvers for Boundary Value Problems of Ordinary Differential Equations, R package version 1.2.4 (2013).
URL http://CRAN.R-project.org/package=bvpSolve

[20] F. Mazzia, J. R. Cash, K. Soetaert, Solving boundary value problems in the open source software R : Package **bvpSolve**, Opuscula Mathematica,in press. doi:10.7494/OpMath.

[21] J. R. Cash, F. Mazzia, A new mesh selection algorithm, based on conditioning, for two-point boundary value codes, J. Comput. Appl. Math. 184 (2) (2005) 362–381. doi:10.1016/j.cam.2005.01.016.
URL http://dx.doi.org/10.1016/j.cam.2005.01.016

[22] J. R. Cash, F. Mazzia, Hybrid mesh selection algorithms based on conditioning for two-point boundary value problems, JNAIAM J. Numer. Anal. Ind. Appl. Math. 1 (1) (2006) 81–90.

[23] R. H. Miller, An experimental method for testing numerical stability in Initial-Value Problems, Journal of Computational Physics 2 (1967) 1–7.

[24] F. Mazzia, A. M. Nagy, Stiffness detection strategy for explicit Runge Kutta methods, AIP Conference Proceedings 1281 (1) (2010) 239–242. doi:10.1063/1.3498435.
URL http://link.aip.org/link/?APC/1281/239/1

[25] F. Mazzia, C. Magherini, Test Set for Initial Value Problem Solvers, release 2.4, Department of Mathematics, University of Bari and INdAM, Research Unit of Bari (February 2008).
URL http://www.dm.uniba.it/~testset

[26] U. M. Ascher, R. M. M. Mattheij, R. D. Russell, Numerical solution of boundary value problems for ordinary differential equations, Vol. 13 of Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995, corrected reprint of the 1988 original.

[27] J. R. Cash, A. H. Karp, A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides, ACM Transactions on Mathematical Software 16 (3) (1990) 201–222.
URL http://doi.acm.org/10.1145/79505.79507

[28] K. Soetaert, J. R. Cash, F. Mazzia, **deTestSet**: Testset for differential equations, R package version 1.1.1 (2013).
URL http://CRAN.R-project.org/package=deSolve

[29] F. Mazzia, J. R. Cash, K. Soetaert, A test set for stiff initial value problem solvers in the open source software R : Package **deTestSet**, J. Comput. Appl. Math. 236 (16) (2012) 4119–4131. doi:10.1016/j.cam.2012.03.014.
URL http://dx.doi.org/10.1016/j.cam.2012.03.014

[30] K. Soetaert, J. Cash, F. Mazzia, Solving differential equations in R, Use R!, Springer, New York, 2012. doi:10.1007/978-3-642-28070-2.
URL http://dx.doi.org/10.1007/978-3-642-28070-2

[31] C. Moler, MATLAB News & Notes, Stiff Differential Equations (May 2003).
URL http://www.mathworks.com/company/newsletters/news_notes/clevescorner/may03_cleve.html