Stiffness detection strategy for Explicit Runge Kutta Methods[1]

F. Mazzia* and A. M. Nagy*

* *Dipartimento di Matematica, Università di Bari, Via Orabona 4, 70125 Bari, Italy*

Abstract. This paper describes a new practical strategy to detect stiffness based on explicit Runge-Kutta schemes. This strategy implements an operative definition of stiffness based on the computation of two conditioning parameters. Test results, using a modified version of the MATLAB code DOPRI5, indicate that the new strategy is able to detect whether a problem could be solved more efficiently by an implicit method.

Keywords: Ordinary Differential Equations, Stiffness, Initial Value Problems.
PACS: 65L05,65L10,65L50

INTRODUCTION

We consider initial value problems for systems of ordinary differential equations

$$y' = f(t,y), \quad t_0 \le t \le t_f, \quad y(t_0) = y_0; \tag{1}$$

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), informations about the behavior of the solution are required. In general, some problems are referred to as "stiff", and many authors call the explicit numerical methods as non stiff methods and the implicit ones as stiff methods, by relating the term stiff to the definition given, for example, in [3]: *Stiff equations are problems for which explicit methods don't work.* In [2] the evolution of the notion of stiffness has been reviewed and a precise definition which encompasses all the previous ones has been presented for uniformly asymptotically stable problems. We note that the needs of applications, especially those arising in the construction of robust and general purpose codes, require nowadays a formally precise definition of stiffness.

Early stiffness detection is important when we use explicit methods, to have the information that the problem could be solved more efficiently using an implicit scheme. Many researchers have attempted to find a suitable way to automatically detect stiffness for explicit numerical schemes. For example in [5] 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 order $err = O(h^p), \tilde{err} = O(h^q)$, with $q < h$, usually $err < \tilde{err}$, if the stepsize is limited by stability requirements and $\tilde{err} \gg err$ when the stepsize is limited by accuracy requirements. More details about this procedures are described in [3], pag. 21 where a second possibility to detect stiffness based on an approximation of the dominant eigenvalues is reported. This technique has been implemented in the code dopri5 [3]. In the following we recall the definition of stiffness presented in [1, 2, 4] and we describe a new stiffness detection algorithm for explicit Runge-Kutta methods.

STIFFNESS DETECTION FOR EXPLICIT RUNGE-KUTTA METHODS

We consider the initial value problem

$$\begin{aligned} y' &= A(t)y + q(t), \qquad t \in [t_0, t_f] \\ y(0) &= y_0, \end{aligned} \tag{2}$$

[1] Work developed within the project "Numerical methods and software for differential equations".

We recall that a solution $y(t)$ is called *asymptotically uniformly stable* if, for all $\varepsilon > 0$, there is a $\delta > 0$ such that any other solution $\hat{y}(x)$ of (2) satisfying $\|y(t_1) - \hat{y}(t_1)\| \leq \delta$ at some points $t_1 \geq t_0$, also satisfies $\|y(t) - \hat{y}(t)\| \leq \varepsilon$ for all $t > t_1$ and, in addition $\|y(t) - \hat{y}(t)\| \to 0$ as $t \to \infty$.
Suppose that in (2) we perturbed the initial value y_0. The corresponding perturbed ODE is:

$$\begin{aligned} \hat{y}' &= A(t)\hat{y} + q(t), \qquad t \in [t_0, t_f] \\ \hat{y}(0) &= y_0 + \eta, \end{aligned} \tag{3}$$

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

$$\begin{aligned} z' &= A(t)z, \qquad t \in [t_0, t_f] \\ z(0) &= \eta, \end{aligned} \tag{4}$$

For the solution of (4) (see, e.g., [4]), we can introduce two parameters, related to the conditioning of the problem, as follows:

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

with $\delta > 0$, $\|\eta\| \neq 0$.

Definition: (See [1]) The initial value problem (1) is called stiff in $[t_0, t_0 + T]$, $T > 0$ if

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

If the conditioning parameters $\kappa_c(t_0, t_f)$ and $\gamma_c(t_0, t_f)$ are of moderate values, then the problem is said to be well conditioned; for large values of $\kappa_c(t_0, t_f)$ the problem is ill-conditioned. The stiffness ratio $\sigma_c(T)$ is used to determine the stiffness of the problems. In order to detect stiffness using $\sigma_c(T)$, we need to compute a discrete approximation of the conditioning parameters. In the following we describe the implementation in the MATLAB version of the code DOPRI5, even if the algorithm could be implemented in every Runge-Kutta code. Usually a numerical code computes an approximation of y, on the grid $\pi = \{t_0, t_1, ..., t_N\}$ with grid spacing $h_i = t_i - t_{i-1}$, $i = 1, ..., N$. In addition to this, we also compute an approximation of $\hat{y}$, the solution of (3), on the same grid. This allows us to approximate the conditioning parameters as follows:

$$\kappa_\eta(\pi) = \frac{1}{\|\eta\|_2} \max_{i=1,...,N} (\|z_i\|_2), \tag{7}$$

$$\begin{aligned} &\hat{\gamma}_\eta(\pi) = \frac{1}{\|\eta\|_2} \frac{1}{(t_N - t_0)} \sum_{i=1}^{N} h_i \max(\|z_i\|_2, \|z_{i-1}\|_2), \\ &\bar{\gamma}_\eta(\pi) = \frac{1}{\|\eta\|_2} \frac{1}{(t_N - t_0)} \sum_{i=1}^{N} \frac{h_i}{2} (\|z_i\|_2 + \|z_{i-1}\|_2), \end{aligned} \tag{8}$$

where $z_i = \hat{y}_i - y_i$, and two approximations of the stiffness ratio as follows:

$$\hat{\sigma}_\eta(\pi) = \frac{\kappa_\eta(\pi)}{\hat{\gamma}_\eta(\pi)}, \quad \bar{\sigma}_\eta(\pi) = \frac{\kappa_\eta(\pi)}{\bar{\gamma}_\eta(\pi)}. \tag{9}$$

It is important to detect the value of η that yields a good approximation of $\sigma_c(T)$. After some extensive numerical experiments we choose η as the dominant eigenvector of $A(t)$, with $t \approx t_0$. This eigenvector is computed by using the technique described in [3] as follows

$$Y_i = y(t_0) + h \sum_{j=1}^{i-1} b_{ij} k_j, \qquad i = 2, 3, ..., 7,$$

$$\eta = sc * \frac{Y_7 - Y_6}{\|Y_7 - Y_6\|_\infty},$$

and we choose the scaling factor $sc = rtol * \|y_0\|$ if $\|y_0\| > 0$ and $sc = atol$ when $\|y_0\| = 0$.
In order to control the local error the code DOPRI5 uses, at each step, a standard relative error estimation on y,

$$e_y(h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{E_i(h)}{sc}\right)^2},$$

where $sc = Atol_i + |y_i| * Rtol_i$ and $E(h)$ is an approximation of the local error. To have a good approximation of $\hat{y}$ and z, we check the errors on these solutions using the same strategy used for y.

This generate a new mesh selection strategy:

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

The step is accepted if $e_t(h) < 1$ and $\sigma_{diff} = \frac{|\bar{\sigma}_\eta - \hat{\sigma}_\eta|}{\max(1,\sigma_\eta)} < r_\sigma = 0.5$. In the new algorithm at each step we doubled the number of function evaluations, but this is a very modest price to pay for explaining the behavior of the integration and give information about the conditioning of the problem.

NUMERICAL EXAMPLES

In the following we present some numerical results to demonstrate the performance of the new stiff detection algorithm. Comparisons are made between the original code and the modified one. The results are tabulated in Table 1 for different values of $rtol$ and $atol$. In the table we show the time t of the integration when stiffness is detected. We denote it by t_o for the original code and t_m for the new one (– in the tables means that the code was not able to recognize the stiffness in the problem and * means that the code failed to compute the solution). 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 1.1 *Classical problem due to Robertson [3] which models a chemical reaction. The equations and initial values are given by*

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

Example 1.2 *The Brusselator problem modeled as (for more details see page 6 in [3])*

$$\begin{aligned} &\frac{\partial u}{\partial t} = A + u^2 v - (B+1)u + \alpha\frac{\partial^2 u}{\partial x^2}, \qquad t \in [0,10] \\ &\frac{\partial v}{\partial t} = Bu - uv^2 + \alpha\frac{\partial^2 v}{\partial x^2}, \end{aligned} \tag{11}$$

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 1.3 *The Model of Flame Propagation given by*

$$\begin{aligned} &y' = y^2 - y^3, \quad t \in [0, 2/\delta] \\ &y(0) = \delta. \end{aligned} \tag{12}$$

By looking at Table 1 we see that for the Robertson problem the two codes detect stiffness at a similar time. For $rtol = atol =$ 1e-4, they fail to give a solution and the very big value of κ_η means that the

TABLE 1. Examples 1.1,1.2 and 1.3

Problem		t_o	t_m	rtol	atol	$\hat{\sigma}_\eta$	κ_η
Robertson	$t_f = 10.$	– (*)	0.10505 (*)	1e-4	1e-4	4.63e12	1.38e11
		0.02661	0.03123	1e-4	1e-7	1.00e2	1.0e0
		0.03159	0.03916	1e-5	1e-8	7.08e2	1.0e0
		0.03354	0.03349	1e-6	1e-10	7.46e3	1.0e0
Brusselator		0.69436	0.33411	8e-2	8e-2	6.13e0	1.0e0
	$N = 40$	0.60002	0.12524	1e-4	1e-7	1.94e1	1.0e0
	$t_f = 10$	–	0.77738	1e-8	1e-8	1.94e1	1.0e0
Brusselator		0.21481	0.08467	8e-2	8e-2	3.92e0	1.0064e0
	$N = 80$	0.15044	0.04052	1e-4	1e-7	4.85e1	1.0e0
	$t_f = 10$	1.4843	0.13567	1e-8	1e-8	3.24e1	1.0e0
Flame propagation	δ = 1e-1	–	–	1e-4	1e-7	3.18e0	1.51e1
	δ = 1e-2	192.7248	184.5933	1e-4	1e-7	2.98e1	1.50e3
	δ = 1e-3	1074.1924	1021.9909	1e-4	1e-7	2.91e2	1.47e5
	δ = 1e-4	10084.9898	10020.9922	1e-4	1e-7	2.89e3	1.45e7

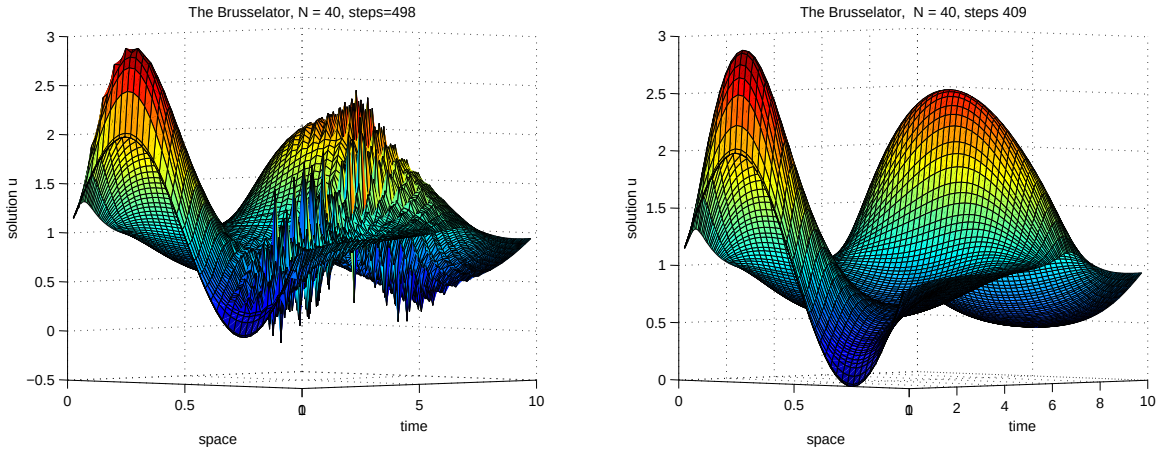

FIGURE 1. The numerical solution for equation (11) using the original (left) and the modified DOPRI5 code (right).

numerical approximation is unstable. The value of $\kappa_\eta = 1$ test out to be the same conditioning parameter as the continuous one. 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 an approximate relative error e_y smaller than $5.2 \cdot 10^{-5}$. Moreover, for $N = 80$, the stiffness ratio increases and the new method detect stiffness at the beginning of the time interval. For the third problem, the stiffness depends on the value of δ. For $\delta = 0.1$ the problem is not stiff, the value of $\kappa_\eta > 1$ 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.

REFERENCES

1. L. Brugnano, D. Trigiante, On the characterization of stiffness for ODEs. *Dynamics of Continuous, Discrete and Impulsive Systems* **2** (1996) 317–335.
2. L. Brugnano, F. Mazzia, D. Trigiante. Fifty Years of Stiffness. *Annals of the European Academy of Sciences*, 2009 (in press) arXiv:0910.3780.
3. E. Hairer, & G. Wanner *Solving ordinary differential equations. II. Stiff and differential algebraic problems*, 2nd edn. Springer. 1996
4. F. Iavernaro, F. Mazzia, D. Trigiante. Stability and conditioning in Numerical Analysis. *JNAIAM* **1** (2006) 91–112.
5. L. F. Shampine. *Stiffness and nonstiff differential equation solvers, II: Detection stiffness with Runge-Kutta methods.* ACM Trans. Math. Software **3**(1), 1977, pp. 44–53.