1.1.5 Stability of explicit methods

The concept of stability refers to the evolution of relative global errors (cf. 1.4) during the run of a numerical scheme for a fixed (finite) step-size h. Stability requires that the relative global error remains bounded.

Reliable estimates for the relative global error may be difficult or impossible – in many cases where differential equations are solved numerically there is barely knowledge about "true" values of the solution. Even if some analysis leading to estimates on the bounds of the relative global error is possible, it is highly probable that this is very specific for the given differential equation (i.e. the right hand side function f and maybe the initial conditions). Such an analysis thus would explore what is called inherent stability or instability.

A common approach to explore stability of a numerical method is to restrict the analysis to a simple linear model of constant growth called Dahlquist equation:

$$\boxed{y' = Ay \quad y(0) = 1} \tag{1.14}$$

with $A \in C$ or A a constant matrix in the case that $y = \vec{y}$ is vector-valued.

Of course, the system (1.14) is very simple and its exact solutions are easily found to be $y = e^{Ax}$ [1]. For $A = \operatorname{Re} A + \underbrace{\operatorname{Im} A}_{\omega} i \in C$ the solution is an oscillation:

$$\boxed{y = e^{\operatorname{Re} A x}\left(\underbrace{\cos(\omega x) + i \sin(\omega x)}_{= e^{i\omega x}}\right)} \tag{1.14a}$$

with exponential real amplitude function $e^{\operatorname{Re} A x}$.

A numerical scheme and the boundedness of relative errors are analysed with respect to (1.14). This will lead to the concept of absolute stability.

Example 1.11: Absolute stability analysis for the Heun-method

The Heun-method is a Runge-Kutta method of order 2 with 2 stages (cf. 1.9a):

$$y_0 = y(x_0) = y(0) = 1$$

$$y_{k+1} = y_k + \frac{1}{2} h\big(f(x_k, y_k) + f(x_k + h, y_k + hf(x_k, y_k))\big)$$

Applied to the Dahlquist system (1.14) with its simple r.h.s. function $f(x, y) = Ay$ the recursive scheme above simplifies to

$$y_0 = 1$$

$$y_{k+1} = y_k + \frac{1}{2} h\big(Ay_k + A(y_k + hAy_k)\big) = \underbrace{\left(1 + hA + \frac{1}{2}(hA)^2\right)}_{\text{Function } F(hA) = F(z)} y_k \tag{1.15}$$

---

[1] In the case that $y = \vec{y}$ Is vector-valued the solution is given by the matrix exponential function defined by the matrix power series $\vec{y} = \left(e^{xA}\right) = \sum_{k=0}^{\infty} \frac{x^k}{k!} A^k$.

In condensed form (1.15) reads as $y_{k+1} = F(z)y_k$ with $F(z) = 1 + z + \frac{1}{2}z^2$ and $z := hA$. The function $F(z)$ is called linear stability function.

Another representation then is $\boxed{y_{k+1} = \left(|F(z)|e^{i\cdot\arg F(z)}\right)y_k}$. **(1.15a)**

Three cases are to be distinguished:

(1) ReA < 0: The solution's amplitude (cf. 1.14a) is exponentially decaying. So it is required by stability that the approximate values $y_k \quad (k = 0,1,2,...)$ in (1.15) exponentially decay, too. But this implies that $|F(z)| < 1$.

(2) ReA = 0: The solution's amplitude (cf. 1.14a) is constantly 1. So it is required by stability that the approximate values $y_k \quad (k = 0,1,2,...)$ in (1.15) have the same constant amplitude, too. But this implies that $|F(z)| = 1$.

(3) ReA > 0: The solution's amplitude (cf. 1.14a) is exponentially increasing. So it is required by stability that the approximate values $y_k \quad (k = 0,1,2,...)$ in (1.15) exponentially increase, too. But this implies that $|F(z)| > 1$.

Cases (2 and (3) are always true assuming that ImA = 0[1], only the first case is critical as is shown by the following contour plot of the complex region $\{z \in C \mid |F(z)| < 1\}$.

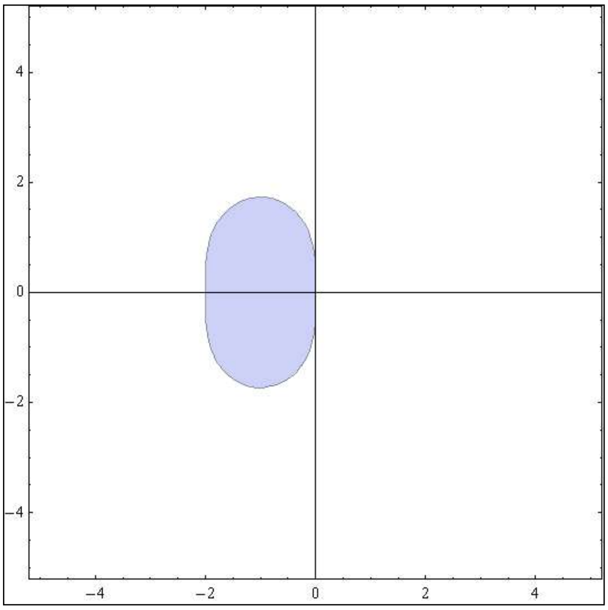

Figure 1.8: Region of absolute stability $\{z \in C \mid |F(z)| < 1\}$ for the Heun-method.

For negative values of A it is required that

-2 < hReA < 0 **.**

This constitutes a stability constraint on the step-size h.

Obviously, the region $\{z \in C \mid |F(z)| < 1\}$ does not intersect the right half plane $\{z \in C \mid \mathrm{Re}\, z \geq 0\}$. So the cases (2) and (3) are always true and there are no stability constraints on the step-size in these cases assuming that ImA = 0.

The example above is representative for any explicit Runge-Kutta method (cf. 1.11c) and the next theorem is only a matter of generalization.

[1] This means that there are no frequencies in the solution. If ImA ≠ 0 then case (2) is unstable because $|F(z)| > 1$ and case (1) is unstable if $|F(z)| \geq 1$, i.e. ImA is too high (frequency !). Note that $z = h\,\mathrm{Re}\,A + h\,j\,\mathrm{Im}\,A$.

Theorem 1.3: Linear stability polynomial for explicit Runge-Kutta methods[1]

The linear stability function for an explicit s-stage Runge-Kutta method with corresponding Butcher tableau (cf. 1.11c) is a polynomial depending on the a-numbers and b-numbers:

$$F(z) = 1 + b_1 k_1(z) + b_2 k_2(z) + \cdots + b_s k_s(z) \tag{1.16a}$$

With recursively defined polynomials

$$\begin{aligned} &k_1(z) = z \\ &k_{j+1}(z) = z\left(1 + a_{j+1,1}k_1(z) + a_{j+1,2}k_2(z) + a_{j+1,3}k_3(z) + \cdots + a_{j+1,j}k_j(z)\right) \end{aligned} \tag{1.16b}$$

for $j = 1,2,\ldots,s-1$.

(Proof: By definition of an s-stage Runge-Kutta method $y_{k+1} = y_k + hb_1k_1 + hb_2k_2 + \cdots + hb_sk_s$. Using the special r.h.s. function $f(x,y) = Ay$ the following computations are straightforward:

$$k_1 = Ay_k \Rightarrow hk_1 = z\, y_k =: k_1(z) y_k$$
$$k_2 = A\left(y_k + ha_{2,1}k_1\right) = A\left(y_k + ha_{2,1}Ay_k\right) \Rightarrow hk_2 = z(1 + a_{2,1}z)\, y_k = z(1 + a_{2,1}k_1(z)) =: k_2(z) y_k$$
$$k_3 = A\left(y_k + h\left(a_{3,1}k_1 + a_{3,2}k_2\right)\right) = A\left(y_k + h\left(a_{3,1}Ay_k + a_{3,2}A\left(y_k + ha_{2,1}Ay_k\right)\right)\right) \Rightarrow$$
$$hk_3 = z(1 + z(a_{3,1} + a_{3,2}(1 + a_{2,1}z)))\, y_k = z(1 + a_{3,1}k_1(z) + a_{3,2}k_2(z))\, y_k =: k_3(z) y_k$$

This shows (1.16b) for the first three stages. The general proof is straightforward, too:

$$k_{j+1} = f(x_k, y_k + h(a_{j+1,1}k_1 + a_{j+1,2}k_2 + \cdots + a_{j+1,j}k_j)) = A\left(y_k + h\left(a_{j+1,1}k_1 + a_{j+1,2}k_2 + \cdots + a_{j+1,j}k_j\right)\right)$$
$$= A\left(y_k + a_{j+1,1}\underbrace{hk_1}_{=:k_1(z)y_k} + a_{j+1,2}\underbrace{hk_2}_{=:k_2(z)y_k} + \cdots + a_{j+1,j}\underbrace{hk_j}_{=:k_j(z)y_k}\right)$$
$$= Ay_k + a_{j+1,1}Ak_1(z)y_k + a_{j+1,2}Ak_2(z)y_k + \cdots + a_{j+1,j}Ak_j(z)y_k \Rightarrow$$

$$hk_{j+1} = \underbrace{Ah}_{=:z} y_k + a_{j+1,1}\underbrace{Ah}_{=:z} k_1(z)y_k + a_{j+1,2}Ahk_2(z)y_k + \cdots + a_{j+1,j}Ahk_j(z)y_k$$
$$= \left(z + a_{j+1,1}zk_1(z) + a_{j+1,2}zk_2(z) + \cdots + a_{j+1,j}zk_j(z)\right) y_k$$
$$= \underbrace{z\left(1 + a_{j+1,1}k_1(z) + a_{j+1,2}k_2(z) + \cdots + a_{j+1,j}k_j(z)\right)}_{=:k_{j+1}(z)} y_k$$

This proves (1.16b)

■)

[1] The linear stability function of a general (not necessarily explicit) Runge-Kutta method (cf. 1.11ab) is given by the formula $\dfrac{\det(I - zA + z\, j|\cdot b)}{\det(I - zA)}$. Here A denotes the matrix of a-number, b the vector of b-numbers and j is the vector consisting of 1s. This expression generally is rational in z. But in the case of an explicit method with strictly lower triangular A-matrix it is polynomial because the denominator is 1 (the matrix $I - zA$ is lower triangular with 1s in the diagonal).

A method with linear stability function F is defined to be <u>A-stable</u> if the region of absolute stability $\{z \in C \mid |F(z)| < 1\}$ includes the left complex half plane $\{z \in C \mid \operatorname{Re} z < 0\}$.

Corollary 1.1: Explicit Runge-Kutta methods are not A-stable

(<u>Proof</u>: By Theorem 1.3 the linear stability function F of an explicit Runge-Kutta method is a polynomial. As a polynomial of degree > 0 F is unbounded for x < 0 (on the negative x-axis) because $\lim_{x\to-\infty} |F(x)| = \infty$ ■)

Example 1.12: Linear stability functions and regions of absolute stability for Bogacki-Shampine-Ralston 3(2)

The Butcher tableaus are:

0				
1/2	1/2			
3/4	0	3/4		
1	2/9	1/3	4/9	
	2/9	1/3	4/9	0
	7/24	1/4	1/3	1/8

0				
1/2	1/2			
3/4	0	3/4		
1	2/9	1/3	4/9	
	2/9	1/3	4/9	0
	-5/72	1/12	1/9	-1/8

According to Theorem 1.3 (1.16b):

$$k_1(z) = z$$

$$k_2(z) = z(1 + \frac{1}{2}z)$$

$$k_3(z) = z(1 + 0z + \frac{3}{4}z(1 + \frac{1}{2}z))$$

$$k_4(z) = z(1 + \frac{2}{9}z + \frac{1}{3}z(1 + \frac{1}{2}z) + \frac{4}{9}z(1 + 0z + \frac{3}{4}z(1 + \frac{1}{2}z)))$$

For the first rows of b-numbers (of order 3) in the r.h.s. tableau:

$$F_1(z) = 1 + \frac{2}{9}k_1(z) + \frac{1}{3}k_2(z) + \frac{4}{9}k_3(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6}$$

and for the second (embedded) row:

$$F_2(z) = 1 - \frac{5}{72}k_1(z) + \frac{1}{12}k_2(z) + \frac{1}{9}k_3(z) - \frac{1}{8}k_4(z) = 1 - \frac{z^3}{48} - \frac{z^4}{48}.$$

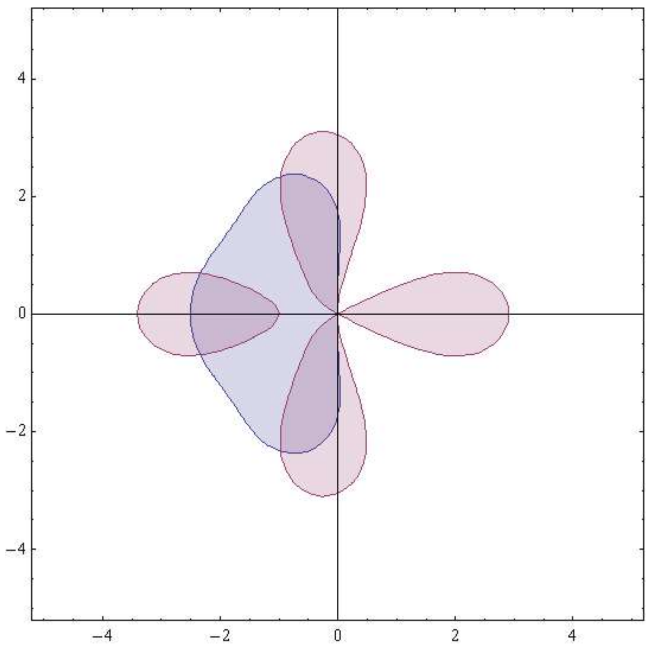

Figure 1.9: Regions of absolute stability $\{z \in C \mid |F_{1,2}(z)| < 1\}$ for the BSR3(2) method.

For negative values of $\mathrm{Re}A$ it is required that

-2.51275< $h\mathrm{Re}A$ < 0.

These conditions constitute stability constraints on the step-size h in the first method.

Example 1.13: Linear stability functions and regions of absolute stability for some widely used Runge-Kutta methods

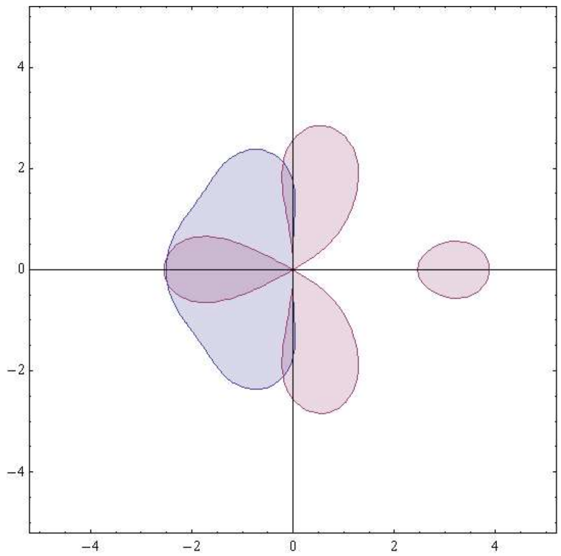

Figure 1.10a: Sofroniou-Spaletta method SS3(2)

$$F_1(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6}$$

$$F_2(z) = 1 + \frac{1}{288}(8 + \sqrt{82})z^3 + \frac{1}{288}(-16 + \sqrt{82})z^4$$

For negative values of $\mathrm{Re}A$ it is required that

-2.51275< $h\mathrm{Re}A$ < 0.

These conditions constitute stability constraints on the step-size h in the first method.

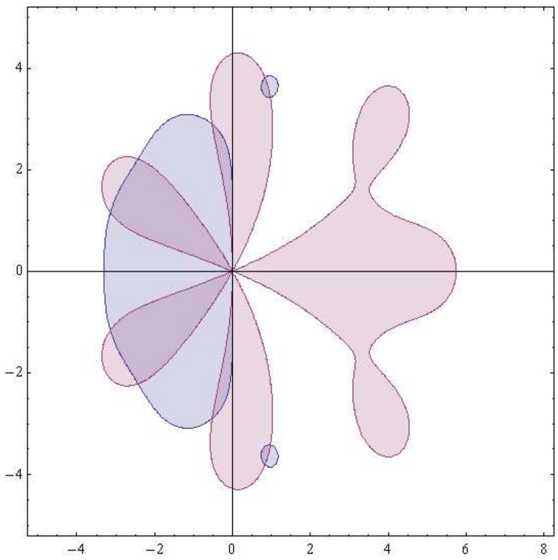

Figure 1.10b: Dormand-Prince 5(4)

$$F_1(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \frac{z^5}{120} + \frac{z^6}{600}$$

$$F_2(z) = 1 - \frac{97z^5}{120000} + \frac{13z^5}{40000} - \frac{z^7}{24000}$$

For negative values of $\mathrm{Re}A$ it is required that

$-3.30657 < h\mathrm{Re}A < 0$.

These conditions constitute stability constraints on the step-size h in the first method.

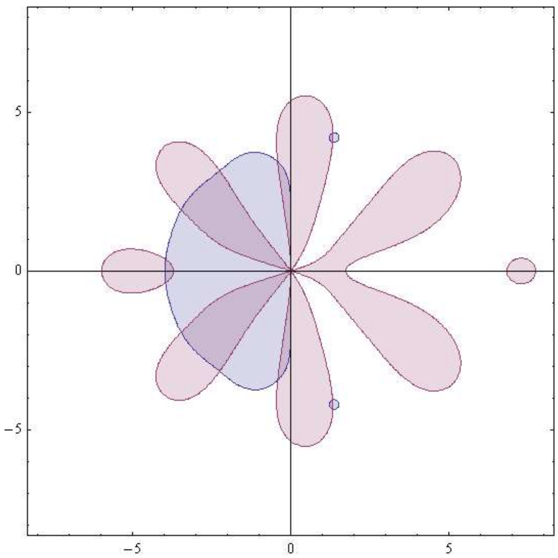

Figure 1.10c: Bogacki-Shampine BS5(4)

$$1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \frac{z^5}{120} + \frac{17291z^6}{12418560} + \frac{269z^7}{1379840}$$

$$1 - \frac{65\,477\,z^5}{1\,286\,568\,360} + \frac{4\,415\,797\,z^6}{192\,127\,541\,760} + \frac{38\,728\,973\,z^7}{6\,916\,591\,503\,360} - \frac{885\,817\,z^8}{768\,510\,167\,040}$$

For negative values of $\mathrm{Re}A$ it is required that

$-3.98793 < h\mathrm{Re}A < 0$.

These conditions constitute stability constraints on the step-size h in the first method.

1.1.6 Stiffness

The phenomenon of stiffness is a serious challenge for numerical solvers for differential equations because it normally has negative impacts on the stability of a numerical scheme and enforces very small step-sizes leading to high computational costs – especially with explicit methods. Handling stiffness in a numerical scheme is a rather complex task: It requires that stiffness or non-stiffness (!) is detected and that the numerical solver may switch between more appropriate methods depending on the results of stiffness detection[1] or non-stiffness detection on runtime.

Stiffness occurs often in engineering sciences when a solution function of a differential equation is a composition (e.g., a superposition) of terms with strongly distinctive scales. Typical examples of such scales are frequencies and damping factors in amplitudes. Often, the scale is a timing scale and the solution has oscillatory components for the short time scale that are damped away in the long time scale. In many cases stiffness is a transient phenomenon.

Example 1.14: A simple one-dimensional equation leading to stiffness

The linear inhomogenuous equation $y' = -100y + 99e^{-x}$ has the solutions

$y = ce^{-100x} + e^{-x} \quad (c \in R)$ and particularly $y = e^{-x} - e^{-100x}$ for the initial condition $y(0) = 0$.

x_k	Classical Runge-Kutta	ExplicitEuler	Reference	Exact
0.	3.41061×10^{-13}	-1.19349×10^{-15}	0.	0.
0.1	-289.993	9.9	0.904792	0.904792
0.2	-84 650.4	-80.1421	0.818731	0.818731
0.3	-2.46335×10^{7}	729.384	0.740818	0.740818
0.4	-7.16835×10^{9}	-6557.13	0.67032	0.67032
0.5	-2.08599×10^{12}	59 020.8	0.606531	0.606531
0.6	-6.07023×10^{14}	-531 181.	0.548812	0.548812
0.7	-1.76644×10^{17}	4.78063×10^{6}	0.496585	0.496585
0.8	-5.14033×10^{19}	-4.30257×10^{7}	0.449329	0.449329
0.9	-1.49584×10^{22}	3.87231×10^{8}	0.40657	0.40657
1.	-4.35288×10^{24}	-3.48508×10^{9}	0.367879	0.367879
1.1	-1.26669×10^{27}	3.13657×10^{10}	0.332871	0.332871
1.2	-3.68606×10^{29}	-2.82292×10^{11}	0.301194	0.301194
1.3	-1.07264×10^{32}	2.54062×10^{12}	0.272532	0.272532
1.4	-3.1214×10^{34}	-2.28656×10^{13}	0.246597	0.246597
1.5	-9.08326×10^{36}	2.05791×10^{14}	0.22313	0.22313
1.6	-2.64323×10^{39}	-1.85212×10^{15}	0.201897	0.201897
1.7	-7.6918×10^{41}	1.6669×10^{16}	0.182684	0.182684
1.8	-2.23831×10^{44}	-1.50021×10^{17}	0.165299	0.165299
1.9	-6.51349×10^{46}	1.35019×10^{18}	0.149569	0.149569

x_k	Classical Runge-Kutta	ExplicitEuler	Reference	Exact
1.	0.367879	0.367879	0.367879	0.367879
1.1	0.370437	0.331091	0.332871	0.332871
1.2	11.2669	0.3156	0.301194	0.301194
1.3	3191.32	0.14142	0.272532	0.272532
1.4	928 596.	1.42528	0.246597	0.246597
1.5	2.70221×10^{8}	-10.3862	0.22313	0.22313
1.6	7.86344×10^{10}	95.685	0.201897	0.201897
1.7	2.28826×10^{13}	-859.166	0.182684	0.182684
1.8	6.65884×10^{15}	7734.3	0.165299	0.165299
1.9	1.93772×10^{18}	-69 607.1	0.149569	0.149569
2.	5.63877×10^{20}	626 465.	0.135335	0.135335
2.1	1.64088×10^{23}	-5.63819×10^{6}	0.122456	0.122456
2.2	4.77497×10^{25}	5.07437×10^{7}	0.110803	0.110803
2.3	1.38952×10^{28}	-4.56693×10^{8}	0.100259	0.100259
2.4	4.04349×10^{30}	4.11024×10^{9}	0.090718	0.090718
2.5	1.17666×10^{33}	-3.69921×10^{10}	0.082085	0.082085
2.6	3.42407×10^{35}	3.32929×10^{11}	0.0742736	0.0742736
2.7	9.96404×10^{37}	-2.99636×10^{12}	0.0672055	0.0672055
2.8	2.89954×10^{40}	2.69673×10^{13}	0.0608101	0.0608101
2.9	8.43765×10^{42}	-2.42705×10^{14}	0.0550232	0.0550232

Tables 1.7ab: Failure of classical Runge-Kutta method for fixed step-size h = 0.1.

(The reference solution is computed with working precision 32 and stiffness switching algorithms.)

The second attempt was made with initial condition redefined to $y(1) = e^{-1} - e^{-100}$.

The rather large partial derivative

$f_y = -100$ takes the role of $\mathrm{Re}A$ in the absolute stability analysis (cf. Example 1.11).

The linear stability function of the classical Runge-Kutta method is:

$$F(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}$$

and the boundary of the region of absolute stability intersects the negative real x-axis at -2.78529 and 0. From this the condition $-2.78529 < -100h < 0 \Leftrightarrow 0 < h < \frac{2.78529}{100} = 0.027859$ is deduced, and obviously,

[1] Cf. http://reference.wolfram.com/mathematica/tutorial/NDSolveStiffnessTest.html for a real expert reference.

the step-size h = 0.1 is far beyond this range. For the explicit Euler method a similar computation leads to

$$-2 < -100h < 0 \Leftrightarrow 0 < h < 0.02 .$$

The next tables show attempts with h = 0.015, h = 0.02 and h = 0.025 starting from initial coordinates $(x = 1, y = e^{-1} - e^{-100})$.

Tables 1.7cd:

h = 0.015

x_k	Classical Runge-Kutta	ExplicitEuler	Reference	Exact
1.	0.367879	0.367879	0.367879	0.367879
1.015	0.362405	0.362361	0.362402	0.362402
1.03	0.357011	0.356987	0.357007	0.357007
1.045	0.351696	0.351662	0.351692	0.351692
1.06	0.34646	0.346431	0.346456	0.346456
1.075	0.341302	0.341271	0.341298	0.341298
1.09	0.33622	0.336192	0.336216	0.336216
1.105	0.331215	0.331186	0.331211	0.331211
1.12	0.326283	0.326255	0.32628	0.32628
1.135	0.321426	0.321398	0.321422	0.321422

x_k	Classical Runge-Kutta	ExplicitEuler	Reference	Exact
19.87	2.34733×10^{-9}	2.34712×10^{-9}	2.3473×10^{-9}	2.3473×10^{-9}
19.885	2.31238×10^{-9}	2.31218×10^{-9}	2.31235×10^{-9}	2.31235×10^{-9}
19.9	2.27795×10^{-9}	2.27776×10^{-9}	2.27793×10^{-9}	2.27793×10^{-9}
19.915	2.24404×10^{-9}	2.24384×10^{-9}	2.24401×10^{-9}	2.24401×10^{-9}
19.93	2.21063×10^{-9}	2.21044×10^{-9}	2.2106×10^{-9}	2.2106×10^{-9}
19.945	2.17772×10^{-9}	2.17753×10^{-9}	2.17769×10^{-9}	2.17769×10^{-9}
19.96	2.14529×10^{-9}	2.14511×10^{-9}	2.14527×10^{-9}	2.14527×10^{-9}
19.975	2.11336×10^{-9}	2.11317×10^{-9}	2.11333×10^{-9}	2.11333×10^{-9}
19.9875	2.08709×10^{-9}	2.08696×10^{-9}	2.08708×10^{-9}	2.08708×10^{-9}
20.	2.06116×10^{-9}	2.06102×10^{-9}	2.06115×10^{-9}	2.06115×10^{-9}

Tables 1.7ef:

h = 0.02

x_k	Classical Runge-Kutta	ExplicitEuler	Reference	Exact
1.	0.367879	0.367879	0.367879	0.367879
1.02	0.360607	0.360522	0.360595	0.360595
1.04	0.353471	0.353456	0.353455	0.353455
1.06	0.346473	0.346384	0.346456	0.346456
1.08	0.339613	0.339598	0.339596	0.339596
1.1	0.332888	0.332801	0.332871	0.332871
1.12	0.326296	0.326284	0.32628	0.32628
1.14	0.319835	0.31975	0.319819	0.319819
1.16	0.313502	0.313492	0.313486	0.313486
1.18	0.307294	0.307211	0.307279	0.307279

x_k	Classical Runge-Kutta	ExplicitEuler	Reference	Exact
19.82	2.46778×10^{-9}	-0.0000369069	2.46765×10^{-9}	2.46765×10^{-9}
19.84	2.41891×10^{-9}	0.0000369118	2.41879×10^{-9}	2.41879×10^{-9}
19.86	2.37101×10^{-9}	-0.000036907	2.37089×10^{-9}	2.37089×10^{-9}
19.88	2.32406×10^{-9}	0.0000369117	2.32394×10^{-9}	2.32394×10^{-9}
19.9	2.27804×10^{-9}	-0.0000369071	2.27793×10^{-9}	2.27793×10^{-9}
19.92	2.23294×10^{-9}	0.0000369116	2.23282×10^{-9}	2.23282×10^{-9}
19.94	2.18872×10^{-9}	-0.0000369072	2.18861×10^{-9}	2.18861×10^{-9}
19.96	2.14538×10^{-9}	0.0000369115	2.14527×10^{-9}	2.14527×10^{-9}
19.98	2.1029×10^{-9}	-0.0000369072	2.10279×10^{-9}	2.10279×10^{-9}
20.	2.06126×10^{-9}	0.0000369114	2.06115×10^{-9}	2.06115×10^{-9}

Tables 1.7gh:

h = 0.025

x_k	Classical Runge-Kutta	ExplicitEuler	Reference	Exact
1.	0.367879	0.367879	0.367879	0.367879
1.025	0.358834	0.358682	0.358796	0.358796
1.05	0.349998	0.349998	0.349938	0.349938
1.075	0.341372	0.3411	0.341298	0.341298
1.1	0.332954	0.333063	0.332871	0.332871
1.125	0.32474	0.324262	0.324652	0.324652
1.15	0.316726	0.317122	0.316637	0.316637
1.175	0.308909	0.307993	0.308819	0.308819
1.2	0.301284	0.302337	0.301194	0.301194
1.225	0.293846	0.29195	0.293758	0.293758

x_k	Classical Runge-Kutta	ExplicitEuler	Reference	Exact
19.775	2.58203×10^{-9}	-8.08816×10^{127}	2.58123×10^{-9}	2.58123×10^{-9}
19.8	2.51828×10^{-9}	1.21322×10^{128}	2.5175×10^{-9}	2.5175×10^{-9}
19.825	2.4561×10^{-9}	-1.81984×10^{128}	2.45534×10^{-9}	2.45534×10^{-9}
19.85	2.39546×10^{-9}	2.72976×10^{128}	2.39472×10^{-9}	2.39472×10^{-9}
19.875	2.33631×10^{-9}	-4.09463×10^{128}	2.33559×10^{-9}	2.33559×10^{-9}
19.9	2.27863×10^{-9}	6.14195×10^{128}	2.27793×10^{-9}	2.27793×10^{-9}
19.925	2.22237×10^{-9}	-9.21292×10^{128}	2.22168×10^{-9}	2.22168×10^{-9}
19.95	2.1675×10^{-9}	1.38194×10^{129}	2.16683×10^{-9}	2.16683×10^{-9}
19.975	2.11398×10^{-9}	-2.07291×10^{129}	2.11333×10^{-9}	2.11333×10^{-9}
20.	2.06179×10^{-9}	3.10936×10^{129}	2.06115×10^{-9}	2.06115×10^{-9}

Example 1.15: A simple one-dimensional stiff equation (Example 1.14 continued)

The tables below show comparisons of different adaptive methods with various accuracy and precision goals. The *Mathematica* method SS3(2) (Sofroniou-Spaletta) looks rather expensive, but the reason is that stiffness detection is off. SS3(2) is more suitable for stiffness detection and would be left if stiffness detection would be on (automatic stiffness switching, cf. Table 1.8c).

Method	Steps	Costs	Error
BSR32	{758, 506}	3794	0.000159676
SS32	{759, 554}	3941	0.000035481

Method	Steps	Costs	Error
BSR32	{1349, 289}	4916	3.43865×10^{-10}
SS32	{2220, 281}	7505	8.58037×10^{-10}

Method	Steps	Costs	Error
BSR32	{113 329, 42 624}	467 861	6.34952×10^{-16}
SS32	{828 666, 187 280}	3 047 840	6.58427×10^{-17}

Table 1.8a: Comparisons of BSR3(2) and SS3(2) with accuracy goal = 4|8|16 and precision goal = 4|8|16.

Method	Steps	Costs	Error
DP54	{885, 2}	5324	3.45653×10^{-10}
BS54	{613, 201}	5700	2.17317×10^{-7}

Method	Steps	Costs	Error
DP54	{23 466, 2634}	156 602	1.42702×10^{-17}
BS54	{15 345, 737}	112 576	1.31396×10^{-16}

Table 1.8b: Comparisons of DP5(4) and BS5(4) with accuracy goal = 8|16 and precision goal = 8|16.

Method	Steps	Costs	Error
BSR32	{931, 15}	2840	2.58261×10^{-9}
SS32	{1805, 0}	5417	1.15295×10^{-9}

Table 1.8c: Comparison of BSR3(2) and SS3(2) with accuracy goal = 8 and precision goal = 8 in the x-interval [1, 9.52509]. At x = 9.52509 stiffness was detected by SS3(2). The additional costs for SS3(2) are partly due to the stiffness detection that is on.

1.1.6.1 Stiffness detection

Detection of stiffness or non-stiffness (!) on runtime generally is not an easy task but in the case of explicit Runge-Kutta methods an elegant and inexpensive approach for stiffness detection is possible. Recalling the definition of explicit Runge-Kutta methods (cf. 11ab) for the last two k-values

$$k_{s-1} = f(x + c_{s-1}h, \underbrace{y + h\sum_{j=1}^{s} a_{s-1,j}k_j}_{g_{s-1}}) \qquad k_s = f(x + c_s h, \underbrace{y + h\sum_{j=1}^{s} a_{s,j}k_j}_{g_s})$$

the formula

$$\boxed{\tilde{\lambda} = \frac{\|k_s - k_{s-1}\|}{\|g_s - g_{s-1}\|}} \qquad \textbf{(1.17)}$$

gives a good estimate for the partial derivative $f_y := \frac{\partial f}{\partial y}$ at the position $x+h$ provided that $c_{s-1} = c_s = 1$.

The value $\tilde{\lambda}$ takes the role of $|\mathrm{Re}A|$ [1] in the absolute stability analysis (cf. Example 1.11). By testing $|h\tilde{\lambda}|$ against the absolute boundary values of the absolute stability region stiffness can be detected.

Example 1.16: Stiffness detection with Sofroniou-Spaletta method SS3(2)

From Example 1.13 and Figure 1.10a the linear stability function of SS3(2) is known to be $F_1(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6}$ with boundary values on the negative x-axis equal to {-2.51275, 0}.

The stiffnest test in this case is $|h\tilde{\lambda}| \le s|-2.51275|$. The number s is a safety factor (by default 0.8).[2]

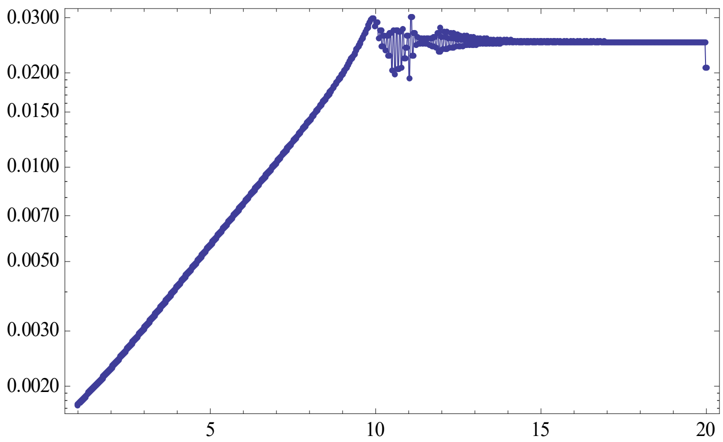

Figure 1.11: Step data plot for Example 1.16 with accuracy and precision goals equal to 8

Since $|\tilde{\lambda}| \approx |f_y| = 100$ the stiffness test produces $h \le 0.0251275$.

The step data plot is compatible with Table 1.8c: Stiffness is detected at x = 9.52509. Above this x-value oscillations of the step-size begin and continue on if the method SS3(2) is kept running.

Stiffness is not an inherent property of a differential equation. It is a relative concept and the occurrence of stiffness generally depends on the numerical method and the associated evolution of the step-size. Stiffness occurs when the step-size h is such that $|h\tilde{\lambda}|$ is near or beyond the bounds of the absolute stability condition.

---

[1] For vector differential equations generalizing (1.1) to $\vec{y}'(x) = \vec{f}(x, \vec{y}(x)) = \left(f_1(x, \vec{y}(x)), f_1(x, \vec{y}(x)), \cdots, f_m(x, \vec{y}(x))\right)$ with initial vector condition $\vec{y}'(x_0) = \vec{y}_0$ the value $\tilde{\lambda}$ is an absolute estimate for the dominant eigenvalue of the Jacobian matrix $J_f = \frac{\partial(f_1, f_2, \cdots, f_m)}{\partial(y_1, y_2, \cdots, y_m)}$. Generally, estimates of dominant eigenvalues are rather expensive and are based on "advanced" methods of linear algebra (cf. http://reference.wolfram.com/mathematica/tutorial/NDSolveStiffnessTest.html).

[2] In modern implementations it is useful to specify the maximum number of successive and total times that the stiffness test is allowed to fail by a pair of number called maximal repetitions. Typical values are {3,5}.