4 Stiffness and Stability

In addition to having a stable *problem*, i.e., a problem for which small changes in the initial conditions elicit only small changes in the solution, there are two basic notions of *numerical* stability. The first notion of stability is concerned with the behavior of the numerical solution for a fixed value $t > 0$ as $h \to 0$.

Definition 4.1 *A numerical IVP solver is* zero stable *if small perturbations in the initial conditions do not cause the numerical approximation to diverge away from the true solution provided the true solution of the initial value problem is bounded.*

For a consistent s-step method one can show that the notion of stability and the fact that its characteristic polynomial ρ satisfies the root condition are equivalent. Therefore, as mentioned earlier, for an s-step method we have

$$\text{convergence} \quad \Longleftrightarrow \quad \text{consistent \& stable.}$$

This concept of stability also plays an important role in determining the global truncation error. In fact, for a convergent (consistent and stable) method the local truncation errors add up as expected, i.e., a convergent s-step method with $\mathcal{O}(h^{p+1})$ local truncation error has a global error of order $\mathcal{O}(h^p)$.

4.1 Linear Stability Analysis

A second notion of stability is concerned with the behavior of the solution as $t \to \infty$ with a fixed stepsize h. This notion of stability is often referred to as *absolute stability*, and it is important when dealing with *stiff* ODEs. An *absolutely stable* numerical method is one for which the numerical solution of a stable problem behaves also in this controlled fashion.

A Model Problem: For $\lambda \in \mathbb{R}$ we consider the family of scalar *linear* initial value problems (the discussion in the book by Iserles uses the analogous system case)

$$\begin{aligned} y'(t) &= \lambda y(t), \quad t \in [0, T], \\ y(0) &= y_0. \end{aligned}$$

The solution of this problem is given by

$$y(t) = y_0 e^{\lambda t}.$$

Now we take the same differential equation, but with perturbed initial condition

$$y_\delta(0) = y_0 + \delta.$$

Then the general solution still is $y_\delta(t) = ce^{\lambda t}$. However, the initial condition now implies

$$y_\delta(t) = (y_0 + \delta)e^{\lambda t}.$$

Therefore, if $\lambda \leq 0$, a small change in the initial condition causes only a small change in the solution and therefore the problem is a *stable problem*. However, if $\lambda > 0$, then

large changes in the solution will occur (even for small perturbations of the initial condition), and the *problem is unstable*. Therefore, when studying the stability of numerical methods we will consider the model problem for $\lambda \leq 0$ only. Even though we will study only stability with respect to the model problem, it can be shown that the results of this analysis also apply to other linear (and some nonlinear) problems.

Example We study how Euler's method behaves for the stable model problem above, i.e., in the case $\lambda \leq 0$. Since $f(t, y) = \lambda y(t)$ Euler's method states that

$$\begin{aligned} y_{n+1} &= y_n + hf(t_n, y_n) \\ &= y_n + h\lambda y_n \\ &= (1 + \lambda h) y_n. \end{aligned}$$

Therefore, by induction,

$$y_n = (1 + \lambda h)^n y_0.$$

Since the exact problem has an exponentially decaying solution for $\lambda < 0$, a stable numerical method should exhibit the same behavior. Therefore, in order to ensure stability of Euler's method we need that the so-called *growth factor* $|1 + \lambda h| < 1$. For real $\lambda < 0$ this is equivalent to

$$-2 < h\lambda < 0 \quad \Longleftrightarrow \quad h < \frac{-2}{\lambda}.$$

Thus, Euler's method is only *conditionally stable*, i.e., the step size has to be chosen sufficiently small to ensure stability.

The set of λh for which the growth factor is less than one is called the *linear stability domain* $\mathcal{D}$ (or *region of absolute stability*).

Example For Euler's method we have

$$|1 + \lambda h| < 1$$

so that (for complex λ)

$$\mathcal{D}_{Euler} = \{z = \lambda h \in \mathbb{C} : \ |z + 1| < 1\},$$

a rather small circular subset of the left half of the complex plane.

FIGURE

Example On the other hand, we can show that the implicit or *backward Euler* method

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

is *unconditionally stable* for the above problem.

To see this we have

$$y_{n+1} = y_n + h\lambda y_{n+1}$$

or

$$y_{n+1} = \frac{1}{1 - \lambda h} y_n.$$

Therefore,

$$y_n = \left(\frac{1}{1-\lambda h}\right)^n y_0.$$

Now, for (real) $\lambda < 0$, the growth factor

$$\left(\frac{1}{1-\lambda h}\right) < 1$$

for any $h > 0$, and we can choose the step size h arbitrarily large.

Of course, this statement pertains only to the stability of the method. In order to achieve an appropriate accuracy, h still has to be chosen reasonably small. However, as we will see below, we do not have to worry about stepsize constraints imposed by the *stiffness* of the problem.

If we allow complex λ, then the linear stability domain for the backward Euler method is given by the entire negative complex half-plane, i.e.,

$$\mathcal{D}_{backwardEuler} = \{z = \lambda h \in \mathbb{C} : \ \mathrm{Re} z < 0\} = \mathbb{C}^-.$$

In general, absolute stability of a linear multistep formula can be determined with the help of its characteristic polynomials. In fact, an s-step method is absolutely stable if the roots of the polynomial $\phi = \rho - \lambda h \sigma$ lie inside the unit disk. Here ρ and σ are defined as earlier, i.e.,

$$\begin{aligned} \rho(w) &= \sum_{m=0}^{s} a_m w^m \\ \sigma(w) &= \sum_{m=0}^{s} b_m w^m \end{aligned}$$

and a_m and b_m are the coefficients of the s-step method

$$\sum_{m=0}^{s} a_m y_{n+m} = h \sum_{m=0}^{s} b_m f(t_{n+m}, y_{n+m})$$

with $f(t_{n+m}, y_{n+m}) = \lambda y_{n+m}$, $m = 0, \ldots, s$, given by the model problem.

The linear stability domain of an s-step method is then the region in the complex plane for which the roots of the polynomial $\phi = \rho - \lambda h \sigma$ lie inside the unit disk.

Example For Euler's method $\rho(w) = w - 1$, and $\sigma(w) = 1$, so that

$$\phi(w) = \rho(w) - \lambda h \sigma(w) = (w-1) - \lambda h = w - (1 + \lambda h)$$

with root $1 + \lambda h$. The region of absolute stability $\mathcal{D}$ is therefore

$$\mathcal{D}_{Euler} = \{z = \lambda h \in \mathbb{C} : \ |1 + z| < 1\},$$

the same region we found earlier.

4.2 Stiffness

The phenomenon of *stiffness* is not precisely defined in the literature. Some attempts at describing a stiff problem are:

- A problem is stiff if it contains widely varying time scales, i.e., some components of the solution decay much more rapidly than others.
- A problem is stiff if the stepsize is dictated by stability requirements rather than by accuracy requirements.
- A problem is stiff if explicit methods don't work, or work only extremely slowly.
- A linear problem is stiff if all of its eigenvalues have negative real part, and the *stiffness ratio* (the ratio of the magnitudes of the real parts of the largest and smallest eigenvalues) is large.
- More generally, a problem is stiff if the eigenvalues of the Jacobian of $\boldsymbol{f}$ differ greatly in magnitude.

Example A stiff ODE is illustrated in the Matlab script `StiffDemo2.m`. If one lights a match, the ball of flame grows rapidly until it reaches critical size. Then it remains at that size because the amount of oxygen being consumed by the combustion in the interior of the ball balances the amount available through the surface. The model is

$$y'(t) = y^2(t) - y^3(t), \quad y(0) = \delta, \quad t \in [0, 2/\delta].$$

The scalar quantity $y(t)$ represents the radius of the ball of flame at time t. The y^2 and y^3 terms in the model come from the surface area and volume, respectively. The critical parameter is the initial radius, δ, which is "small". The solution is sought on an interval of time inversely proportional to δ. The solution to this problem will grow very slowly until t reaches $1/\delta$, then a quick transition occurs to a value close to 1, where the solution then remains. The exact solution is given by

$$y(t) = \frac{1}{W(ae^{a-t}) + 1},$$

where W is the *Lambert W* function (the solution of the equation $W(z)e^{W(z)} = z$) and $a = 1/\delta - 1$ (see the Maple worksheet `StiffDemo2.mws`).

Note how it takes the "non-stiff" solver `ode45` — an adaptive explicit fourth-fifth order Runge-Kutta method — in `StiffDemo2.m` longer and longer to obtain a solution for decreasing values of δ. The stiff solver `ode23s` is an adaptive second-third order Rosenbrock method.

This problem is initially not stiff, but becomes so as the solution approaches the steady state of 1.

Remark

1. Stiff ODEs arise in various applications; e.g., when modeling chemical reactions, in control theory, or electrical circuits, such as the *Van der Pol equation* in relaxation oscillation (see the Matlab script `VanderPolDemo.m`).
2. One way to deal with a stiff problem is to have an *unconditionally stable* solver.

4.3 *A*-Stability

Often unconditional stability, in fact, unconditional stability for the model problem

$$\begin{aligned} y'(t) &= \lambda y(t), \quad t \in [0,T] \\ y(0) &= y_0 \end{aligned}$$

is all that is needed for an effective stiff solver. This property is usually called *A-stability*.

Definition 4.2 *A numerical IVP solver is* A-stable *if its region of absolute stability includes the entire complex half-plane with negative real part,* $\mathbb{C}^-$.

We already have seen one A-stable method earlier: the backward (or implicit) Euler method

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}).$$

In general, only implicit methods are candidates for stiff solvers. In particular, a corollary to Lemma 4.2 in the Iserles book states that no explicit Runge-Kutta can be A-stable. The case of s-step methods is covered in the book by Iserles in the form of Lemmas 4.7 and 4.8. Moreover, the following theorem (Dahlquist's Second Barrier) reveals the limited accuracy that can be achieved by A-stable s-step methods.

Theorem 4.3 *If a linear s-step method is A-stable then it must be an implicit method. Moreover, the order of the method is at most 2.*

Theorem 4.3 implies that the only other s-step Adams method we need to consider is the *implicit trapezoid method* (or second-order Adams-Moulton method) introduced earlier:

$$\boldsymbol{y}_{n+2} = \boldsymbol{y}_{n+1} + \frac{h}{2}\left[\boldsymbol{f}(t_{n+2}, \boldsymbol{y}_{n+2}) + \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})\right]. \tag{48}$$

To see that the implicit trapezoid method is A-stable we consider

$$\boldsymbol{y}_{n+2} = \boldsymbol{y}_{n+1} + \frac{h}{2}\lambda\left[\boldsymbol{y}_{n+2} + \boldsymbol{y}_{n+1}\right]$$

or

$$(1 - \lambda\frac{h}{2})\boldsymbol{y}_{n+2} = (1 + \lambda\frac{h}{2})\boldsymbol{y}_{n+1}.$$

Therefore, as can be verified easily by induction,

$$\boldsymbol{y}_n = \left(\frac{1 + \lambda\frac{h}{2}}{1 - \lambda\frac{h}{2}}\right)^n \boldsymbol{y}_0,$$

and, for $\lambda < 0$, the growth factor

$$\frac{1 + \lambda\frac{h}{2}}{1 - \lambda\frac{h}{2}} = \frac{2 + \lambda h}{2 - \lambda h} < 1$$

for any $h > 0$. In other words, the linear stability domain for the trapezoidal rule is

$$\mathcal{D}_{TR} = \{z = \lambda h \in \mathbb{C} : \ \mathrm{Re}z < 0\} = \mathbb{C}^-.$$

Another method admitted by the Second Dahlquist Barrier is the second-order BDF method. It can also be shown to be A-stable.

As mentioned above, explicit Runge-Kutta methods cannot be A-stable. However, all implict Gauss-Legendre (Runge-Kutta) methods — such as the implicit midpoint rule — are A-stable.

Remark Of course, one can also try to develop methods that do not completely satisfy the condition of A-stability, and hope to achieve higher order by doing this. The state-of-the-art stiff solvers today fall into one of the following two categories:

1. Higher-order BDF methods are almost A-stable (see Figure 4.4 in the Iserles book). This kind of stability is referred to as $A(\alpha)$-stability, where α indicates the angle of diversion from the vertical imaginary axis. *Gear methods* are based on such BDF methods, and achieve higher-order along with numerical stability for stiff problems by monitoring the largest and smallest eigenvalues of the Jacobian matrix, and thus assessing and dealing with the stiffness adaptively. Gear methods employ variable order as well as variable step size. In MATLAB, Gear methods are implemented in the stiff solver `ode15s`. In Maple some of these methods are available through the `method=lsode` option (invoking the Livermore ODEPACK by Hindmarsh) to `dsolve,numeric`.

2. Another class of stiff solvers are so-called *Rosenbrock* methods (which are related to implicit Runge-Kutta methods). In MATLAB a second-third order Rosenbrock method is implemented in the routine `ode23s`. In Maple a third-fourth order Rosenbrock method is the default implementation in `dsolve,numeric` with the `stiff=true` option.

References for this subject are the books "Numerical Initial Value Problems in Ordinary Differential Equations" by Bill Gear (1971), or "Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems" by Hairer and Wanner (1991), or the two books "Numerical Solution of Ordinary Differential Equations" by Larry Shampine (1994) and "Computer Solution of Ordinary Differential Equations: the Initial Value Problem" by Shampine and Gordon (1975).