

## 6. Analysis of Convergence

The procedures outlined above can be used to derive various finite difference algorithms suited to ODE's. There arises naturally the question of which algorithms are better than others or whether they work for all cases. We would like an algorithm to give an approximation to the exact solution of the problem and to get better results as we increase the resolution of the algorithm by decreasing the step size  $k$ . Suppose we are seeking the solution to (2.1) for  $t \in [0, T]$  through an algorithm  $\mathcal{A}$  that produces the approximations  $\{Q^n \cong q(t^n)\}_{n=0,1,\dots,N}$   $t^n = nk$ ,  $k = T/N$ . At each step we expect there to be an error  $E^n = Q^n - q(t^n)$ . If we use exact arithmetic then  $E^0 = 0$  since we just apply the initial condition, i.e.  $Q^0 = q_0 \implies E^0 = 0$ . If we use computer arithmetic however there might be an unavoidable initial error, e.g. when  $q_0$  is irrational.

DEFINITION 2. An algorithm  $\mathcal{A}$  to solve the IVP (2.1) is **convergent** if

$$(6.1) \quad \lim_{\substack{k \rightarrow 0 \\ nk=T}} E^n = 0$$

This just states that the error should go to zero as we decrease the step size but keep the product of step size and number of steps constant, i.e.  $nk = T$ .

We now ask whether commonly used algorithms are convergent. Since it is easier to work on simple cases first we start with the simplest algorithms and problems first. We know from ODE theory that linear equations can be solved analytically in closed form and one would expect that any numerical algorithm should be able to approximate the solution to any desired degree of precision also. Therefore we first study algorithms that solve the linear IVP

$$(6.2) \quad \begin{cases} q' = \lambda q + \sigma \\ q(t=0) = q_0 \end{cases},$$

where  $\sigma(t)$  is a source function. We shall call (6.2) the *model problem*. In plain language (6.2) states that the change in  $q$  is given by two effects. The first effect is proportional to  $q$  with a constant of proportionality  $\lambda$  while the second is specified by the source function  $\sigma$ . The analytical solution to this problem is given by

$$(6.3) \quad q(t) = e^{\lambda t} q_0 + \int_0^t e^{\lambda(t-\tau)} \sigma(\tau) d\tau$$

which is known as the *Duhamel principle*. Its interpretation is quite educational. The first term shows that the initial condition gets amplified by  $e^{\lambda t}$ . The function  $e^{\lambda t}$  is known as the *propagator function* since it transfers the initial condition forward to time  $t$ . Source terms also get propagated forward in time by the same function but over a time interval reflecting the difference between the time  $t$  and the time  $\tau$  when the source was applied. There is a cumulation of the effect of source terms as shown by the integral above.

**6.1. Convergence of the forward Euler method for the model problem.** Having selected a suitably simple IVP, we now select a simple numerical method, namely Euler's method

$$(6.4) \quad Q^{n+1} = Q^n + k f(Q^n)$$

and study its convergence properties. Euler's method applied to the model problem gives

$$(6.5) \quad Q^{n+1} = (1 + k\lambda)Q^n + k \sigma^n.$$

We must now construct an expression for the error  $E^n$ . Equation (6.5) suggests that we try to establish a relation recurrence between  $E^{n+1}$  and  $E^n$ . A Taylor series expansion of  $q(t^{n+1})$  gives

$$(6.6) \quad q(t^{n+1}) = q(t^n) + kq'(t^n) + \frac{k^2 q''(\xi^n)}{2}$$

with  $\xi^n \in (t^n, t^{n+1})$  and  $q''$  assumed to be bounded. Since  $q'(t^n) = f(t^n, q(t^n))$  we obtain

$$(6.7) \quad q(t^{n+1}) = (1 + k\lambda)q(t^n) + k \sigma^n + \frac{k^2 q''(\xi^n)}{2}.$$

Subtraction of (6.7) from (6.5) leads to the desired recurrence relation between errors

$$(6.8) \quad E^{n+1} = (1 + k\lambda) E^n - \frac{k^2 q''(\xi^n)}{2}.$$

This shows that the error at the previous time step gets amplified by  $1 + k\lambda$  and a new error proportional to  $k^2$  is introduced during this step of the algorithm. The additional error introduced during step  $n$  of the algorithm shall be called the *one-step error*

$$(6.9) \quad \omega^n = \frac{k^2 q''(\xi^n)}{2}.$$

Note that Euler's method is obtained by truncating the forward finite difference series (3.37) to the first term. This approximation leads to an error in the approximation of the derivative at time step  $n$  which shall be called *the truncation error*

$$(6.10) \quad \tau^n = \left( \frac{1}{k} \Delta_+ - \frac{d}{dt} \right) q(t^n)$$

$$(6.11) \quad = \frac{q(t^{n+1}) - q(t^n)}{k} - q'(t^n).$$

Taylor series expansion gives

$$(6.12) \quad \tau^n = \frac{k q''(\xi^n)}{2}$$

and we note that  $\omega^n = k \tau^n$ , i.e. the one-step error is one order higher than the truncation error.

We can repeatedly apply the recurrence relation (6.8)

$$(6.13) \quad E^n = (1 + k\lambda) E^{n-1} - \omega^{n-1}$$

$$(6.14) \quad = (1 + k\lambda) [(1 + k\lambda) E^{n-2} - \omega^{n-2}] - \omega^{n-1}$$

$$(6.15) \quad = (1 + k\lambda)^2 [(1 + k\lambda) E^{n-3} - \omega^{n-3}] - (1 + k\lambda) \omega^{n-2} - \omega^{n-1}$$

$$(6.16) \quad = \dots$$

and arrive at

$$(6.17) \quad E^{(N)} = (1 + k\lambda)^N E^{(0)} - \sum_{j=0}^{N-1} (1 + k\lambda)^{N-1-j} \omega^{(j)} .$$

We explicitly show superscripts by parantheses to avoid confusion with the exponentiation operations also present. Relation (6.17) is also known as the *discrete Duhamel principle* because of the similarity with (6.3). The initial error is amplified by  $(1 + k\lambda)^N$  and the additional error  $\omega^j$  introduced during subsequent time steps is amplified by  $(1 + k\lambda)^{N-1-j}$ . We see that  $(1 + k\lambda)^N$  plays the role of  $e^{\lambda T}$  and is hence called the discrete propagator. Note that the discrete propagator is a first order truncation of the continuous operator as shown by

$$(6.18) \quad e^{\lambda T} = e^{\lambda k N} = (e^{\lambda k})^N = \left(1 + k\lambda + \frac{k^2 \lambda^2}{2} + \dots\right)^N \cong (1 + k\lambda)^N .$$

Assume that we're using exact arithmetic and therefore  $E^0 = 0$ . If we can show that  $(1 + k\lambda)^N$  remains bounded as  $k \rightarrow 0$  and  $kN = T$  then the first term in (6.17) would have a zero limit. Recall that  $e$  can be expressed as a limit

$$(6.19) \quad \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e .$$

The limit arising from the first term in (6.17) is

$$(6.20) \quad \lim_{\substack{k \rightarrow 0 \\ kN=T}} (1 + k\lambda)^N = \lim_{N \rightarrow \infty} \left(1 + \frac{\lambda T}{N}\right)^N = \lim_{n \rightarrow \infty} \left[\left(1 + \frac{\lambda T}{N}\right)^{\frac{N}{\lambda T}}\right]^{\lambda T} = e^{\lambda T}$$

so the first term in (6.17) does indeed have a zero limit since it leads to the multiplication of a finite quantity  $e^{\lambda T}$  by  $E^{(0)} = 0$ . The second term can be bounded by

$$(6.21) \quad \left| \sum_{j=0}^{N-1} (1 + k\lambda)^{N-1-j} \omega^{(j)} \right| \leq \sum_{j=0}^{N-1} |1 + k\lambda|^{N-1-j} |\omega^{(j)}| \leq \sum_{j=0}^{N-1} e^{k|\lambda|(N-1-j)} |\omega^{(j)}|$$

$$(6.22) \quad \leq e^{k|\lambda|N} \sum_{j=0}^{N-1} |\omega^{(j)}| = e^{|\lambda|T} \sum_{j=0}^{N-1} |\omega^{(j)}| .$$

There are  $N$  terms in the sum with  $N = T/k$ . Let  $\|\omega\|_\infty$  be the largest of the  $|\omega^{(j)}|$  encountered in the sum, this being the one-step error at some  $\xi$

$$(6.23) \quad \|\omega\|_\infty = \frac{k^2 |f''(\xi)|}{2}$$

We then obtain

$$(6.24) \quad \left| \sum_{j=0}^{n-1} (1 + k\lambda)^{n-1-j} \omega^{(j)} \right| \leq e^{|\lambda|T} N \|\omega\|_\infty = e^{|\lambda|T} T \frac{k |q''(\xi)|}{2}$$

This last quantity goes to zero as  $k \rightarrow 0$  since  $e^{|\lambda|T}$ ,  $q''$ ,  $T$  are bounded. We have therefore established that

$$(6.25) \quad \lim_{\substack{k \rightarrow 0 \\ kN=T}} E^N = 0$$

and that the forward Euler algorithm is convergent in exact arithmetic.

### 6.2. The effect of inexact arithmetic and truncation error - stability.

The forward Euler method is convergent in exact arithmetic but this does not necessarily hold true when inexact, computer arithmetic is used. We have shown that the first term in (6.17) goes to zero if  $E^0 = 0$

$$(6.26) \quad \lim_{\substack{k \rightarrow 0 \\ kN=T}} (1 + k\lambda)^N E^{(0)} = 0$$

but if  $E^0 \neq 0$  then  $(1 + k\lambda)^N E^{(0)}$  could lead to an error bounded by  $e^{|\lambda|T} E^0$ . Clearly, just establishing convergence for exact arithmetic is not sufficient for practical purposes. We must establish conditions such that inherent computer arithmetic errors can be controlled. We say that algorithms that permit us to control error growth are *stable* while those in which errors grow without bound are *unstable*.

These intuitive definitions of stability must be made more precise. We take our cue from the continuous dependence of the solution to an IVP on the initial conditions. Suppose we could exactly compute the solution to the linear ODE  $q' = \lambda q + \sigma$  with slightly different initial conditions  $\tilde{q}_0 = q_0 + E^0$ . Here  $E^0 \neq 0$  is intended to represent any initial error due to inexact computer arithmetic. From (??) we know that

$$(6.27) \quad |q(t; q_0) - q(t; \tilde{q}_0)| \leq e^{Lt} |q_0 - \tilde{q}_0| .$$

For the model problem this result can be sharpened since we know the exact solution from (6.3), and we have

$$(6.28) \quad |q(t; q_0) - q(t; \tilde{q}_0)| = e^{\lambda t} E^0 .$$

This means that if an initial error  $E^0$  is unavoidable, and all subsequent computations are done exactly, we can expect an error  $E(T) = e^{\lambda T} E^0$  at time  $T$ . Clearly we cannot hope to better this using approximate computations, so we should recognize that algorithms that are capable of reproducing this sort of behavior are good candidates for further study. Such an algorithm should reproduce the error growth of the exact solutions in the limit of  $k \rightarrow 0$  though it might have different behavior when  $k \neq 0$ . Note that this discussion in terms of propagation of an initial error is also applicable to later stages of the algorithm. We can envisage that the truncation error of the algorithm introduces a certain error at each step. We would like this error to be kept under control as the algorithm progresses.

**DEFINITION 3** (intuitive). *An algorithm  $\mathcal{A}$  producing the approximations  $\{Q^n\}_{n=0,1,\dots}$  to the exact solution of  $q' = f(q)$ ,  $q(0) = q_0$  is said to be **zero-stable** if  $|Q^n - q(t^n)|$  is bounded in the limit of  $k \rightarrow 0$ ,  $n = 1, 2, \dots, N = T/k$ .*

**DEFINITION 4** (exact). *Let  $S$  be the vector with  $r$  components of initial errors in an  $r$ -step algorithm that produce the approximations  $\{Q^n\}_{n=r,r+1,\dots}$  to the exact solution of the IVP  $q' = f(q)$ ,  $q(0) = q_0$*

$$(6.29) \quad S = [Q^0 - q(t^0), Q^1 - q(t^1), \dots, Q^{r-1} - q(t^{r-1})] .$$