3. Higher-Order methods for ODEs

As seen in the previous chapter, it is difficult to ~~d~~ use Taylor methods. We need alternative ways of designing high-order methods. We start with a very popular family of high-order methods, the Runge-Kutta (RK) methods.

We will approximate the IVP,

$$u'(t) = F(u(t), t)$$
$$u(0) = u_0 \qquad \text{—} \quad (3.1)$$

~~RK~~ 3.1 : Runge-Kutta2 (RK-2) method

We consider a discretization of the time interval $[0,T]$, as in the previous chapter. Let $t^n = n\Delta t$.

We have;

$$u(t_{n+1}) - u(t_n) = \int_{t^n}^{t^{n+1}} u'(s)\, ds \qquad \text{(Fundamental Theorem of Calculus)}$$

$$= \int_{t^n}^{t^{n+1}} F(u(s), s)\, ds \qquad \text{(From (3.1))}$$

We need to approximate the above integral by a quadrature, we can use mid-point rule, then,

$$\int_{t^n}^{t^{n+1}} F(u(s),s)\, ds \approx \Delta t\, F\left(u\left(t^n + \frac{\Delta t}{2}\right), t^n + \frac{\Delta t}{2}\right)$$

However, we do not know the value of $u\left(t^n + \frac{\Delta t}{2}\right)$ (mid-point value)

As we are approximating, we can perform a further approximation by using forward Euler, ~~to~~ i.e,

$$u\left(t^n + \frac{\Delta t}{2}\right) \approx u(t_n) + \frac{\Delta t}{2} F(u(t_n), t^n)$$

~~using~~ Combining ~~of~~ all the approximations, we can write:

$$Y_1 = U_n$$

$$Y_2 = U_n + \frac{\Delta t}{2} F(Y_1, t^n) \quad \text{— (3.2)}$$

$$U_{n+1} = U_n + \Delta t\, F\left(Y_2, t^n + \frac{\Delta t}{2}\right)$$

(3.2) is termed the Standard 2-stage Runge-Kutta method. 2-stages refer to Y_1, Y_2 as calculated in (3.2).

Note that (3.2) can be rewritten as

$$U_{n+1} = U_n + \Delta t\, F\left(U_n + \frac{\Delta t}{2} F(U_n, t^n),\ t^n + \frac{\Delta t}{2}\right) \quad \text{— (3.3)}$$

and represents a time marching scheme. with

$$U_0 = u_0.$$

3.1.1 Order of accuracy of RK-2.

The order of accuracy of RK-2 method can be ~~b~~ easily determined by the following strategy:

Step 1 → Consider the simplest linear scalar ODE

$$u' = au, \quad \text{with exact solution}$$
$$u(0) = u_0$$

$$u(t) = u_0 e^{at}.$$

Step 2 : Write down the update formula (3.3) in this simple case.

Check that (3.3) reduces to

$$U_{n+1} = \left(1 + a\Delta t + \frac{a^2 \Delta t^2}{2}\right) U_n .$$

Step 3 : Calculate the corresponding one-step error.

$$
\begin{aligned}
L^n := {} & \cancel{(U_{n+1} - u(t_{n+1}))}\ u(t_{n+1}) - U_{n+1} \\
= {} & e^{a\Delta t} u(t_n) - \left(1 + a\Delta t + \frac{a^2 \Delta t^2}{2}\right) u(t_n) \\
= {} & \left(e^{a\Delta t} - 1 - a\Delta t - \frac{a^2\Delta t^2}{2}\right) u(t_n) \\
= {} & \frac{a^3 \Delta t^3}{6} u(t_n) + \mathcal{O}(\Delta t^4) \\
= {} & \mathcal{O}(\Delta t^3)
\end{aligned}
$$

Step 4 : Based on the discussions in the previous section, if $L_n = \mathcal{O}(\Delta t^3)$, then the method, if stable, is second-order accurate !!!

Step 5 : The order of accuracy carries over from the simplest linear scalar ODE to a general ODE of form (3.1).

3.2 RK4 method:

An even higher-order method results from the following construction:

$$Y_1 = U_n$$
$$Y_2 = U_n + \frac{\Delta t}{2} F(Y_1, t^n)$$
$$Y_3 = U_n + \frac{\Delta t}{2} F(Y_2, t^n + \frac{\Delta t}{2}) \qquad \text{--- } (3.4)$$
$$Y_4 = U_n + \Delta t\, F(Y_3, t^n + \Delta t/2)$$
$$U_{n+1} = U_n + \frac{\Delta t}{6}\Big(F(Y_1, t^n) + 2F(Y_2, t^n + \frac{\Delta t}{2}) + 2F(Y_3, t^n + \frac{\Delta t}{2}) + F(Y_4, t^n + \Delta t)\Big)$$

$$U_0 = u_0$$

Note that RK-4 has 4 stages. ~~and~~ ~~se~~ .

3.2.1 Order of accuracy of RK4.

As in the previous case, we consider the scalar linear ODE

$$u' = au$$
$$u(0) = u_0$$

Check that RK4 in this case reduces to.

$$Y_1 = U_n$$
$$Y_2 = U_n + \frac{a\Delta t}{2} U_n$$
$$Y_3 = U_n + \frac{a\Delta t}{2}\left(U_n + \frac{a\Delta t}{2} U_n\right)$$
$$= U_n + \frac{a\Delta t}{2} U_n + \frac{a^2\Delta t^2}{4} U_n$$
$$Y_4 = U_n + a\Delta t\left(U_n + \frac{a\Delta t}{2} U_n + \frac{a^2\Delta t^2}{4} U_n\right)$$
$$= U_n + a\Delta t\, U_n + \frac{a^2\Delta t^2}{2} U_n + \frac{a^3\Delta t^3}{4} U_n$$

$$\therefore\ U_{n+1} = U_n + a\Delta t\, U_n + \frac{a^2\Delta t^2}{2} U_n + \frac{a^3\Delta t^3}{6} U_n + \frac{a^4\Delta t^4}{24} U_n.$$

Thus, the one-step error associated with RK4 is given by,

$$\ell^n = u(t_{n+1}) - \left(1 + a\Delta t + \frac{a^2\Delta t^2}{2} + \frac{a^3\Delta t^3}{6} + \frac{a^4\Delta t^4}{24}\right) u(t_n)$$

$$= \left(e^{a\Delta t} - 1 - a\Delta t - \frac{a^2\Delta t^2}{2} - \frac{a^3\Delta t^3}{6} - \frac{a^4\Delta t^4}{24}\right) u(t_n)$$

$$= O(\Delta t^5)$$

Hence, RK-4 is expected to be fourth-order accurate !!!

3.3: General form of RK methods.

The above two-examples suggest the following form of s-stage RK methods;

$$Y_1 = U_n + \Delta t \sum_{j=1}^{s} a_{1j} F(Y_j, t_n + c_j \Delta t)$$

$$Y_2 = U_n + \Delta t \sum_{j=1}^{s} a_{2j} F(Y_j, t_n + c_j \Delta t)$$

$$\cdots \qquad (3.5).$$

$$\cdots$$

$$Y_s = U_n + \Delta t \sum_{j=1}^{s} a_{sj} F(Y_j, t_n + c_j \Delta t)$$

$$U_{n+1} = U_n + \Delta t \sum_{j=1}^{s} b_j F(Y_j, t_n + c_j \Delta t).$$

Here $\{a_{ij}\}_{i,j=1}^{s}$, $\{b_j\}_{j=1}^{s}$, $\{c_j\}_{j=1}^{s}$ are coefficients that uniquely specify a RK method.

The coefficients are often written in the form of a table, ~~de~~ called the Butcher tableaux;

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\vdots & \vdots & & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s
\end{array} \qquad — \quad (3.6)$$

Examples:

The Butcher tableaux for RK2 method (3.2) is

$$\begin{array}{c|cc}
0 & 0 & 0 \\
1/2 & 1/2 & 0 \\
\hline
 & 0 & 1
\end{array}$$

The Butcher tableaux for RK4 method (3.4) is

$$\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 & 0 \\
1/2 & 0 & 1/2 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}$$

3.5 Consistency Conditions for RK methods:

For a "consistent" RK method (of s-stages) we need some relation between the coefficients.

In particular, we need:

$$\sum_{j=1}^{s} a_{ij} = c_i \quad — \quad (3.10)$$

$$\sum_{j=1}^{s} b_j = 1 \quad — \quad (3.11)$$

The simplest way to see that (3.10, 3.11) have to be satisfied is to consider the scalar non-autonomous ODE

$$\begin{aligned} u'(t) &= F(u(t), t) \\ u(0) &= u_0 \end{aligned} \quad — \quad (3.12)$$

Given a s-stage RK method to approximate (3.12) of the form;

$$Y_i = u_n + \Delta t \sum_{j=1}^{s} a_{ij} F(Y_j, t^n + c_j \Delta t) \quad 1 \le i \le s \quad — (3.13)$$

and

$$u_{n+1} = u_n + \Delta t \sum_{j=1}^{s} b_j F(Y_j, t^n + c_j \Delta t)$$

Now recall from chapter 1 that the non-autonomous system (3.12) can be written as the following autonomous system,

$$\omega = [\omega_1, \omega_2], \quad G = [F(\omega_1, \omega_2), 1].$$

$$\omega' = G(\omega) \quad — \quad (3.14)$$

Check that (3.14) with initial data:
$\omega(t_n) = \omega_n = [u_n, t^n]$ is completely equivalent to (3.12) in the sense that for any time

$$t^{n+1} = t^n + \Delta t,$$

$$\omega_1(t^{n+1}) = u(t^{n+1})$$
$$\omega_2(t^{n+1}) = t^{n+1} = t_n + \Delta t.$$

Now, we can discretize (3.14) with an s-stage Rk method leading to

$$z_u^i = U_n + \Delta t \sum_{j=1}^{s} a_{ij} F(z_u^i, z_t^i)$$
$$z_t^i = t_n + \Delta t \sum_{j=1}^{s} a_{ij} \qquad \text{— (3.15)}$$

and:

$$\omega_1^{n+1} = U_n + \Delta t \sum_{j=1}^{s} b_j F(z_u^i, z_t^i)$$
$$\omega_2^{n+1} = t_n + \Delta t \sum_{j=1}^{s} b_j$$

We require that the values produced by (3.15) agree with those produced by (3.13) i.e,

$$\omega_1^{n+1} = U_{n+1}$$
and $$\omega_2^{n+1} = t^{n+1} = t^n + \Delta t$$

by inspection from (3.15), we see that

$$\sum_{j=1}^{s} b_j = 1 \quad (\text{proving } (3.11)).$$

also we see that: by setting

$$z_t^i = t^n + c_i \Delta t$$

we obtain: $$\sum_{j=1}^{s} a_{ij} = c_i \quad (\text{proving } (3.10)),$$ we automatically obtain that:

$$z_u^i = Y_i \text{ and.}$$

$$\omega_1^{n+1} = U_{n+1}$$

3.6 Examples of RK methods

We consider the two main examples of RK methods,

Explicit Runge-Kutta (RK) methods

If we set $a_{ij} \equiv 0$ if $j \geq i$ $(\forall i)$, then we obtain an explicit RK method as the computation of the i-th stage Y_i only requires information from previous stages $(Y_1, \dots, Y_{i-1})$. So, we can use the following marching scheme;

$$Y_1 \to Y_2 \to Y_3 \cdots Y_{s-1} \to Y_s$$

to calculate all stage values.

In this case, the A matrix in the Butcher tableaux has only lower triangular structure with zero diagonal entries.

Examples include the standard RK2 and RK4 methods as can be readily seen from their Butcher tableaux.

Diagonally implicit RK methods (DIRK) methods

If we set $a_{ij} \equiv 0$ if $j > i$, for all i, j, then we obtain a DIRK method. Here, computing the i-th stage value Y_i requires $\{Y_1, \dots, Y_i\}$. So we need to solve a non-linear equation of the form:

$$Y_i - \Delta t\, a_{ii} F(Y_i, t^n + c_i \Delta t) = U_n + \Delta t \sum_{\substack{j<i \\ j=1}}^{s} a_{ij} F(Y_j, t^n + c_j \Delta t),$$

~~Thus we need to Solve ms no~~

to obtain the i-th stage value.

As a concrete example, we consider the 3-stage second-order accurate DIRK method,

$$Y_1 = U_n$$

$$Y_2 = U_n + \frac{\Delta t}{4}\left[F(Y_1, t^n) + F(Y_2, t_n + \frac{\Delta t}{2})\right]$$

(3.16) — $$Y_3 = U_n + \frac{\Delta t}{3}\left[F(Y_1, t^n) + F(Y_2, t_n + \frac{\Delta t}{2}) + F(Y_3, t^n + \Delta t)\right]$$

$$U_{n+1} = Y_3 = U_n + \frac{\Delta t}{3}\left[F(Y_1, t^n) + F(Y_2, t^n + \frac{\Delta t}{2}) + F(Y_3, t^n + \Delta t)\right]$$

The resulting Butcher tableaux is

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/4 & 1/4 & 0 \\ 1 & 1/3 & 1/3 & 1/3 \\ \hline & 1/3 & 1/3 & 1/3 \end{array}$$

Note that DIRK2 is also termed as TR-BDF2.

3.7: Order of accuracy of general RK methods

In addition to the consistency conditions, we need further conditions on the coefficients. (a_{ij}, c_i, b_i) in order to ensure that a s-stage RK method ( ) has a truncation error of order γ i.e.

$$T_n = O(\Delta t^{\gamma}) \quad \text{or} \quad L_n = O(\Delta t^{\gamma+1})$$

with L_n being the one-step error.

For instance, to obtain second-order methods, we need to impose.

$$\sum_{j=1}^{s} b_j c_j = 1/2$$

Check that both standard RK2 and DIRK2 satisfy these conditions.

For third-order, we also need:

$$\sum_{j=1}^{s} b_j c_j^2 = 1/3 \quad , \quad \sum_{j=1}^{s} \sum_{i=1}^{s} b_i a_{ij} c_j = 1/6$$

~~to~~ More complicated conditions are needed to ensure even ~~more~~ higher order of truncation error. These conditions are outside the scope of this course.

We would just like to mention that we require ~~more than~~ s $(s > r)$ stages to get a truncation error of order r, provided that $r \geq 5$.

$: O(1)$

4. Multi-step methods ~~to~~ for solving ODEs.

The RK methods introduced in the previous. Chapter to numerically solve the ODE IVP,

$$u'(t) = F(u(t), t) \qquad (4.1),$$
$$u(0) = u_0$$

are multi-stage but one-step methods. To see this, observe that ~~only~~ only the value at the previous time step U_n is used to obtain the next time step U_{n+1}.

One possible problem with multi-stage methods such as (3.5) is the fact that a large number of function evaluations are needed, particular if u is a high-dimensional vector and the number of stages (s) is large.

An alternative approach for obtaining high-order methods is to use multi-step methods. The basic idea is to find a value at a time level $t^{n+\gamma}$ i.e. $U_{n+\gamma}$, using the ~~j previous~~ values at the γ previous time steps, namely $U_n, U_{n+1}, \ldots, U_{n+\gamma-1}$.

The simplest form of these methods are the so-called linear multi-step methods given by:

~~$\sum_{j=0}^{\gamma} \alpha_j U_{n+j} = \Delta t \sum_{j=0}^{\gamma} \beta_j F(U_{n+j}, t^{n+j})$~~

$$\sum_{j=0}^{\gamma} \alpha_j U_{n+j} = \Delta t \sum_{j=0}^{\gamma} \beta_j F(U_{n+j}, t^{n+j}) \qquad (4.2)$$

for coefficients $\{\alpha_j, \beta_j\}_{j=1}^{\gamma}$.

Note that this r-step method provides a linear relation between ~~v~~ values at r-time steps.

Special cases of (4.2) are the explicit multi-step method given by the coefficient ~~$\alpha_r = 0$~~ $\beta_r = 0$

The general form (4.2) results in an implicit method.

4.2 Adams-methods:

Adams-methods are special cases of the linear multi-step method (4.2) with coefficients

$$\alpha_r = 1 \quad \alpha_{r-1} = -1, \quad \alpha_j = 0, \quad \forall \ j < r-1.$$

Thus (4.2) takes the form;

$$~~U^{n+r} = U^{n+}~~$$

$$U_{n+r} = U_{n+r-1} + \Delta t \sum_{j=0}^{r} \beta_j F(U_{n+j}, t^{n+j}) \quad -(4.3)$$

we focus on the autonomous case,

$$F(u,t) = f(u)$$

Thus Adams-methods are of the form,

$$U_{n+r} = U_{n+r-1} + \Delta t \sum_{j=0}^{r} \beta_j f(U_{n+j}) \quad - (4.4)$$

4.2.1 Adams-Bashforth Method

The explicit form of (4.4) is given by setting $\beta_r = 0$. Hence, it is of form,

$$U_{n+r} = U_{n+r-1} + \Delta t \sum_{j=0}^{r-1} \beta_j F(U_{n+j}) \quad - (4.5)$$

The coefficients $\{\beta_j\}_{j=0}^{\gamma-1}$ in the Adams-Bashforth method (4.5) are determined to ensure the correct order of accuracy.

One way to do so is by calculating

$$u(t_{n+\gamma}) - u(t_{n+\gamma-1}) = \int_{t_{n+\gamma-1}}^{t_{n+\gamma}} u'(s)\, ds$$

$$= \int_{t^{n+\gamma-1}}^{t_{n+\gamma}} f(u(s))\, ds \qquad \text{———} \quad (4.6)$$

The above integral can be approximated by a quadrature rule, for instance by interpolating $f(u)$ a polynomial $p(t)$ of degree $(\gamma-1)$ by values at $t_n, t_{n+1}, \ldots t_{n+\gamma-1}$. and then integrating the polynomial.

Doing this calculation results in the following methods.

AB 1 $\quad U_{n+1} = U_n + \Delta t\, f(U_n)$ $\quad$ (forward Euler)

AB2 $\quad U_{n+2} = U_{n+1} + \frac{\Delta t}{2}\left(-f(U_n) + 3f(U_{n+1})\right)$

AB3 $\quad U_{n+3} = U_{n+2} + \frac{\Delta t}{12}\left(5f(U_n) - 16f(U_{n+1}) + 23 f(U_{n+2})\right)$

4.2.2 Adams-Moulton methods

The implicit version of the Adams-method (4.4) results by taking $\beta_\gamma \neq 0$.

Here, we repeat the Calculations in (4.6) and approximate $f(u)$ by a polynomial $q(t)$ of degree γ, interpolated from values at points $\{t_n, t_{n+1}, \ldots, t_\gamma\}$

The resulting method is ~~r-th~~ $(r+1)$-th order ~~accu acu~~ accurate. Some examples are given below:

$$\text{AM1} \qquad U_{n+1} = U_n + \frac{\Delta t}{2}\left(f(U_n) + f(U_{n+1})\right) \quad \text{(Trapezoidal rule)}$$

$$\text{AM2} \qquad U_{n+2} = U_{n+1} + \frac{\Delta t}{12}\left(-f(U_n) + 8f(U_{n+1}) + 5f(U_{n+2})\right)$$

$$\text{AM3} \qquad U_{n+3} = U_{n+2} + \frac{\Delta t}{24}\Big(f(U_n) - 5f(U_{n+1}) + 19f(U_{n+2}) + 9f(U_{n+3})\Big)$$

4.3: Truncation error:

The truncation error associated with a linear multi-step method. (4.2) is defined by.

~~$T_{n+r} = \frac{1}{\Delta t}\Big(\sum_{j=0}^{r} \alpha_j U_{n+j} - \Delta t \sum_{j=0}^{r} \beta_j f(u$~~

$$T_{n+r} = \frac{1}{\Delta t}\left(\sum_{j=0}^{r} \alpha_j u(t_{n+j}) - \Delta t \sum_{j=0}^{r} \beta_j f(u(t_{n+j}))\right)$$

$$= \frac{1}{\Delta t}\left(\sum_{j=0}^{r} \alpha_j u(t_{n+j}) - \Delta t \sum_{j=0}^{r} \beta_j u'(t_{n+j})\right) \quad \text{— (4.8)}$$

(from ODE (4.1))

By Taylor expansion;

$$u(t_{n+j}) = u(t_n) + j\Delta t\, u'(t_n) + \frac{j^2\Delta t^2}{2!} u''(t_n) + \cdots + \frac{(j\Delta t)^k}{k!} u^{(k)}(t_n) + \cdots$$

$$u'(t_{n+j}) = u'(t_n) + j\Delta t\, u''(t_n) + \frac{j^2\Delta t^2}{2} u'''(t_n) + \cdots + \frac{(j\Delta t)^k}{k!} u^{(k)}(t_n) + \cdots$$

Substituting into (4.8) and clubbing terms together, we obtain:

$$T_{n+r} = \frac{1}{\Delta t}\left(\sum_{j=0}^{r} \alpha_j\right) u(t_n) + \left(\sum_{j=0}^{r} (j\alpha_j - \beta_j)\right) u'(t_n)$$

$$+ \Delta t \left(\sum_{j=0}^{r}\left(\frac{j^2 \alpha_j}{2} - j\beta_j\right)\right) u''(t_n) + \cdots\cdots$$

$$+ \Delta t^{k-1} \left(\sum_{j=0}^{r}\left(\frac{j^k}{k!}\alpha_j - \frac{j^{k-1}}{(k-1)!}\beta_j\right)\right) u^{(k)}(t_n) + \cdots$$

For consistency we require:

$$\sum_{j=0}^{r} \alpha_j = 0, \qquad \sum_{j=0}^{r} j\alpha_j = \sum_{j=0}^{r} \beta_j$$

A ~~trunct~~ truncation error of order $(\Delta t)^{k}$ is obtained by setting:

$$\sum_{j=0}^{r} \frac{j^q}{q!}\alpha_j = \sum_{j=0}^{r} \frac{j^{q-1}}{(q-1)!}\beta_j$$

$$\forall \quad q \leq k+1 \qquad !!!$$

4.4: Starting values

A r-stage linear multistep method (4.2) needs ~~&~~ r starting values.

$$U_0, U_1, U_2, \ldots U_{r-1}$$

We know $U_0 = u_0$.

How do we specify other starting values?

The usual strategy is Calculate then using a Runge-kutta method of accuracy $(\gamma-1)$ if a γ-stage Adams-Bashforth method is used or RK method of accuracy γ if a γ-stage Adams-Moulton method is used.

The idea of using one-order of accuracy less in the starting values is to use the fact that the one-step error will be of order γ and the total error due to γ-steps (if the method is stable) is $\gamma\, O(\Delta t^{\gamma})$. Thus, the global error remains $O(\Delta t^{\gamma})$.

5. Stability of Numerical methods for ODEs.

Given the ODE IVP,

$$u'(t) = F(u(t), t)$$
$$u(0) = u_0 \qquad \text{— } (5.1)$$

,

all the methods that we have seen so far compute a value U_N at a time level $t^N = N\Delta t = T$.

A method is said to converge if

$$\lim_{\substack{\Delta t \to 0 \\ N\Delta t = T}} U_N = u(T) \qquad \text{— } (5.2)$$

with u being the exact solution of ODE (5.1).

At ~~the very least, a key criteria f~~

For a 1-step method, the starting value coincides with u_0. However, for a r-step multi-step method, we also have to account for starting values, $U_1, U_2, \ldots, U_{r-1}$. We require

$$\lim_{\Delta t \to 0} U_j(\Delta t) = u_0 \; , \quad \forall \; 0 \le j \le r-1 \qquad \text{— } (5.3)$$

Given this, we define

<u>Definition</u> : A numerical method is said to be convergent if the computed solution U_N satisfies (5.2) and (5.3) for every $T > 0$.

Note that convergence, at the very least, is a key requirement for a "good" numerical method.

5.1: Convergence of forward Euler for a linear ODE

We examine the question of convergence in the simplest case.

Consider the linear scalar ODE,

$$u' = au + g(t) \quad — \quad (5.4)$$
$$u(0) = u_0$$

The simplest numerical method for ~~calculati~~ approximating (5.4) is the forward Euler method:

$$\begin{aligned} U_{n+1} &= U_n + \Delta t(aU_n + g(t_n)) \\ &= (1+a\Delta t)U_n + \Delta t\, g(t_n) \end{aligned} \quad — \quad (5.5).$$

$$U_0 = u_0$$

See that (5.3) is automatically satisfied for the one-step method (5.5)

Our aim is to calculate the error defined as

$$E_n := u(t_n) - U_N$$

Recall from Chapter 2 that the truncation error is defined as.

$$T_n := \frac{u(t_{n+1}) - u(t_n)}{\Delta t} - au(t_n) - g(t_n) \quad — \quad (5.6)$$

Check that ~~for (5.5~~ this truncation error is given by

$$T_n = \frac{\Delta t}{2} u''(t_n) + O(\Delta t^3) \quad — \quad \cancel{(5.6)} \quad (5.7)$$

~~Subs We write (5.6) as~~

~~$u(t_{n+1}) = (1+a\Delta t)u(t_n) + \Delta t\, g(t_n) + \Delta t T_n$ — (5.8)~~

~~Substracting (5.5) from (5.8), we obtain~~

We recorrite (5.6) as

$$u(t_{n+1}) = (1+a\,\Delta t)\,u(t_n) + \Delta t\, g(t_n) + \Delta t\, T_n \quad — (5.8)$$

Substracting (5.5) from (5-8), we obtain

$$u(t_{n+1}) - U_{n+1} = (1+a\,\Delta t)\big(u(t_n) - U_n\big) + \Delta t\, T_n \quad — (5.9)$$

using the definition of E_n in (5-9), we have

$$E_{n+1} = (1+a\Delta t)\,E_n + \Delta t\, T_n \quad — (5.10)$$

Thus, the error at a given time level depends on the error at the previous time level and the truncation error.

We iterate (5.10) in the following manner,

$$E_n = (1+a\,\Delta t)\,E_{n-1} + \Delta t\, T_{n-1}$$
$$= (1+a\,\Delta t)\big((1+a\Delta t)\,E_{n-2} + \Delta t\, T_{n-2}\big) + \Delta t\, T_{n-1}$$
$$= (1+a\,\Delta t)^2\,E_{n-2} + \Delta t\,(1+a\,\Delta t)\,T_{n-2} + \Delta t\, T_{n-1}$$

Iterating the above argument N-times, we find

~~$E_n = (1+a\,\Delta t)^n E_0 + \Delta t \sum_{m=1}^{n} (1+a\,\Delta t)^{n-m}\, T_{n-m}$~~

$$E_n = (1+a\Delta t)^n E_0 + \Delta t \sum_{m=1}^{n} (1+a\,\Delta t)^{n-m}\, T_{m-1} \quad — (5.10)$$

The above formula clearly brigs out the contribution of local truncation error to the global error as each step (m+1) contributes on error

$(1+a\Delta t)^{n-m}\, T_{m-1}$ to the global error.

Observe that

$$|1+a\Delta t| \le e^{|a|\Delta t}$$

$$\therefore \quad |1+a\Delta t|^n \le e^{|a|n\Delta t} = e^{|a|T} \quad \text{if } n\Delta t = T$$

Similarly $$(1+a\Delta t)^{n-m} \le e^{|a|\Delta t(n-m)} \le e^{|a|n\Delta t} \le e^{|a|T} \quad \text{——} \quad (5.20)$$

$\therefore$ we bound (5.10) as

$$|E_n| \le |1+a\Delta t|^n |E_0| + \Delta t \sum_{m=1}^{n} |1+a\Delta t|^{n-m} |\tau^{m-1}|$$

$$\le e^{|a|T}\left(|E_0| + \Delta t \sum_{m=1}^{n} \max_m |\tau^{m-1}|\right)$$

let $$\|\tau\|_\infty = \max_{1\le n\le N-1} |\tau^n|$$

From the above, we see that

$$|E_n| \le e^{|a|T}\left(|E_0| + T\|\tau\|_\infty\right)$$

For the forward Euler method:

$$\|\tau\|_\infty \sim \frac{\Delta t}{2}\|u''\|_\infty \sim O(\Delta t)$$

$\therefore$ ~~$|E_n| \le C$~~ as $E_0 = 0$

we see that the error of the forward Euler method is.

$$|E_n| \le Te^{|a|T} O(\Delta t)$$

Hence, $$\lim_{\Delta t\to 0} E_n \to 0$$

thus, Forward Euler satisfies (5.2) and is convergent.

Furthermore, $|E_n| \sim O(\Delta t)$.

Hence, forward Euler is a first-order accurate method !!!

5.2: Convergence of forward Euler method for nonlinear ODEs:

We consider the autonomous ODE:

$$u' = f(u)$$
$$u(0) = u_0 \quad \text{———} \quad (5.11)$$

~~The fo~~ We need to assume that the flux f is Lipshitz Continuous in u (needed for wellposedness)

The forward Euler method to approximate (5.11) is

$$U_{n+1} = U_n + \Delta t\, f(U_n) \quad \text{———} \quad (5.12).$$
$$\cancel{u_0} \quad U_0 = u_0$$

the truncation error is

$$T_n = u(t_{n+1}) - u(t_n). - \Delta t\, f(u(t_n)) \quad \text{—} \quad (5.13)$$

$$\therefore \quad u(t_{n+1}) = u(t_n) + \Delta t\, f(u(t_n)) + T_n. \text{—} \quad (5.14)$$

now subtracting ~~(5.14)~~ (5.12) from (5.14), we ~~obs~~ obtain,

$$u(t_{n+1}) - U_{n+1} = u(t_n) - U_n + \Delta t\left(f(u(t_n)) - f(U_n)\right) + T_n$$

Or $$E_{n+1} = E_n + \Delta t\left(f(u(t_n)) - f(U_n)\right) + T_n$$

$$\therefore \quad \|E_{n+1}\| \leq \|E_n\| + \Delta t\, \|f(u(t_n)) - f(U_n)\| + \|T_n\|$$

(with $\|.\|$ being a vector norm)

as f is Lipschitz continuous,

$$\|f(u(t_n)) - f(U_n)\| \leq L\, \|u(t_n) - U_n\| \leq L\|E_n\|$$

Hence, $$\|E_{n+1}\| \leq (1 + \Delta t L)\, \|E_n\| + \|T_n\|. \quad \text{—} \quad (5.15)$$

Hence, the Error at the n-th time level is bounded by.

~~$\| E_{nrr}$~~ $$\|E_n\| \leq (1+\Delta t L)\|E_{n-1}\| + \|T_{n-1}\|$$

Note that the above is exactly the same formula as we had in page ~~49~~ 48 (except a vector norm and an inequality), So, iterating it ~~n-t~~ n-times we obtain

$$\|E_n\| \leq (1+\Delta t L)^n \|E_0\| + \Delta t \sum_{m=1}^{n} (1+\Delta t L)^{n-m} \|T_{m-1}\|$$

By using the relation. (5-20), we ~~obta~~ see that

$$\|E_n\| \leq e^{LT}\left(\|E_0\| + T\|C\|_\infty\right) \quad — (5.21)$$

with $$\|C\|_\infty = \max_{1\leq m\leq n-1} \|C_m\|$$

as $E_0 \equiv 0$ and we can check that

$$\|C\|_\infty = O(\Delta t)$$

we find that

$$\|E_n\| \leq Te^{LT} O(\Delta t) \longrightarrow 0 \quad \text{as } \Delta t \to +\infty$$

Hence, the forward Euler method is convergent. in the sense of (5.2). for the general ODE (5.1) !!!

5.3 Convergence of consistent one-step methods

It turns out that ~~most~~ explicit one-step methods, such as the Runge-Kutta methods (of chapter 3) can be written in the following form,.

$$U_{n+1} = U_n + \Delta t\, \Phi(U_n, t_n, \Delta t) \quad — (5.22)$$

Consider the 2-stage standard RK method. (3.2).

It can be rewritten in form (3.3);

$$U_{n+1} = U_n + \Delta t\, F\left(U_n + \frac{\Delta t}{2} f(U_n, t_n),\ t_n + \frac{\Delta t}{2}\right)$$

Hence (3.3) can be written in form (5.22) with

$$\Phi(U_n, t_n, \Delta t) = F\left(U_n + \frac{\Delta t}{2} F(U_n, t^n),\ t^n + \frac{\Delta t}{2}\right).$$

We define the method (5.22) to be consistent if

$$\sout{\Phi(U_n, t_n} \qquad \Phi(u, t, 0) = F(u, t).$$

Check that the RK 2 method ~~is~~ is consistent.

We can readily define the truncation error of a consistent 1-step method as;

$$T_n := u(t_{n+1}) - u(t_n) - \Delta t\, \Phi(u(t_n), t_n, \Delta t) \quad \text{—(5.23)}$$

We need to assume that Φ is Lipschitz in u, then.

Subtracting (5.22) from (5.23) we obtain

$$u(t_{n+1}) - U_{n+1} = u(t_n) - U_n + \Delta t\left(\Phi(u(t_n), t_n, \Delta t) - \Phi(U_n, t_n, \Delta t)\right) + \Delta t\, T_n$$

Using Lipshitz continuity

$$\|\Phi(u(t_n), t_n, \Delta t) - \Phi(U_n, t_n, \Delta t)\| \le L\|u(t_n) - U_n\|,$$

we obtain

$$\sout{\|E_n\| \le} \qquad \|E_{n+1}\| \le (1 + \Delta t L)\|E_n\| + \Delta t\|T_n\|.$$

which is identical to (5.15).

Hence, $\|E_n\| \le T e^{LT}\|T_n\|_\infty \to 0$ as $\Delta t \to 0$.

establishing that general Explicit one-step methods like (5.22) are Convergent, if they are consistent !!!

~~5.4~~ . The convergence of multi-step methods is harder to establish and lies outside the ~~so~~ scope of these notes.

5.4 Why Convergence is not enough

Convergence ~~is on~~, in the sense of (5.2) is only necessary for a good method, it is hardly sufficient. We see this fact in an example:

Consider the scalar ODE;

$$u'(t) = -a\left(u - \sin(t)\right) + \cos(t) \qquad \text{— } (5.24)$$
$$u(0) = 0$$

Check that $u(t) = \sin(t)$ is the unique solution of this Scalar linear ODE with any value of a !!!

6 Now, compute the solution of (5.24) with the Forward Euler method with different values of Δt (Number of time levels), we obtain the following error table.

N	Error
100	6.7×10^{66}
200	1.04×10^{59}
300	1.78×10^{5}

The results for ($N = 300$) points is shown in figure (see slide). There are very large oscillations and the solution blows up at these resolutions.

How do we explain this behavior of the method;

~~The reason lies in the fact that~~:

~~(1+~~

Particularly, as we have shown that forward Euler, converges when the error is 3.7×10^{-7}.
$\Delta t \to 0$. Surprisingly, this is still true on: $N = 400$, So, for sufficient small Δt, the method still converges !!! . A clue to this behavior is found in the table

N	Error	$\lvert 1 + a\Delta t \rvert$
200	1.78×10^{5}	1.09
320	2.26×10^{-6}	0.96

Clearly, the value $\lvert 1 + a\Delta t \rvert$ plays a significant role in how the ~~to~~ Error is as ~~as~~ Δt varies.

~~Ab~~ 5.5 Absolute Stability:

The considerations discussed above, give rise to a stronger notion of stability. ~~To~~ This is based on the model equation:

$$u' = au \quad - (5.27)$$
$$u(0) = u_0$$

Note that the solution is ~~$u(t) = u_0 e^{at}$~~ $u(t) = u_0 e^{at}$

Hence, any initial value u_0 will rapidly decay if a is negative.

Applying forward Euler to this problem leads to:

$$U_{n+1} = (1 + a\Delta t)\, U_n \quad - (5.28)$$

We say that ~~S~~ Forward Euler method is absolutely (A-) stable if

$$|1 + a\Delta t| \leq 1 \quad - (5.29)$$

or

$$|U_{n+1}| \leq |U_n| \quad - (5.30)$$

Rem: Note that this notion of stability makes sense when $a \leq 0$ as the exact solution is monotonically decreasing in that case.

From (5.29), it is clear that we need to set Δt such that it holds resulting in:

$$-2 \leq a\Delta t \leq 0$$ (shaded region in figure 5.1).

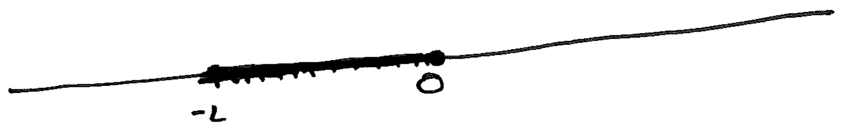


Fig 5.1

Hence, the forward Euler method is A-stable only when the time-step (relative to a) is small enough, justifying our observations in Numerical experiments.

5.5.1 A-stability of Backward Euler:

The backward Euler method applied to (5.27) is

$$\frac{U_{n+1} - U_n}{\Delta t} = a\, U_{n+1}$$

$$\Rightarrow \quad (1 - a\Delta t)\, U_{n+1} = U_n$$

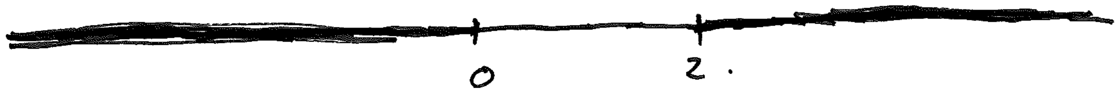
$$\Rightarrow \quad U_{n+1} = \frac{1}{1 - a\Delta t}\, U_n$$

Backward Euler is absolutely stable if (5.30) holds, that is if,

$$\frac{1}{|1 - a\Delta t|} \overset{<}{\cancel{\geq}} 1 \quad \Rightarrow \quad |1 - a\Delta t| \geq 1$$

$$\Rightarrow \quad a\Delta t \in (-\infty, 0] \cup [2, +\infty)$$

So the stability region for backward Euler is plotted in figure 5.2



$a\Delta t$.

Hence, Backward Euler performs so well ~~on the~~ in the numerical experiments.

5.5.2: A-stability of Trapezoidal rule :

Trapezoidal rule applied to (5.27) gives

$$\frac{U_{n+1} - U_n}{\Delta t} = \frac{1}{2}(aU_n + aU_{n+1})$$

$$\left(1 - \frac{a\Delta t}{2}\right) U_{n+1} = \left(1 + \frac{a\Delta t}{2}\right) U_n$$

$$\Rightarrow \quad U_{n+1} = \left(\frac{1 + \frac{a\Delta t}{2}}{1 - \frac{a\Delta t}{2}}\right) U_n$$

Thus Trapezoidal rule will be A-stable if (5.30). is satisfied.

Check that a sufficient condition for doing so if

$$a\Delta t \in (-\infty, 0]$$

5.6. ~~A-stability~~

Stability regions for more complicated Rk and multi-step methods can be computed.

5.6 A-Stability of System of Equations

Consider the linear System of ODEs,

$$u' = Au \quad - \quad (5.31)$$

with the cond assumption that A is diagonalizable (see chap

we recall that,

$$A = R\Lambda R^{-1} \quad \text{with}.$$

$$\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$$

$$R = \left[\, r_1 \,\middle|\, r_2 \,\middle|\, \cdots \,\middle|\, r_m \,\right]$$

with (λ_i, r_i) being the i-th eigenvalue and eigenvector i.e,

$$\cancel{A\lambda} \quad A r_i = \lambda_i r_i \qquad \forall\ 1 \le i \le m$$

As in chapter 2, we rewrite (5-31) as

$$\omega' = \Lambda \omega$$

with $\omega = R^{-1} u$

Thus ~~A~~ (5-31) decouples into m-scalar ODEs of the form

$$\omega_i' = \lambda_i \omega_i \quad \text{— (5-32)}$$

$$1 \le i \le m$$

Applying the forward Euler method to (5-31) results in.

$$U_{n+1} = (I + \Delta t\, A)\, U_n$$ (I is the $(m \times m)$ Identity matrix)

~~$U_{n+1} = U_n + \Delta t\, R \ldots (R^{-1} U_n)$~~

$\Rightarrow$ ~~$(R^{-1} U_{n+1}) = (R^{-1} U_n) + \Delta t$~~

$$U_{n+1} = (I + \Delta t\, R \Lambda R^{-1})\, U_n \,.$$

$$\Rightarrow \quad R^{-1} U_{n+1} = R^{-1} U_n + \Delta t\, \Lambda R^{-1} U_n$$

let $W_n = R^{-1} U_n$

$$\Rightarrow \quad W_{n+1} = W_n + \Delta t\, \Lambda W_n$$

as Λ is a diagonal, this decouples into m-scalar difference equations of form.

$$\omega^i_{n+1} = \omega^i_n + \Delta t\, \lambda_i\, \omega^i_n \quad \text{— (5.33)}$$

with $1 \le i \le m$

Thus absolute ~~A~~-stability of the method boils down to requiring that:

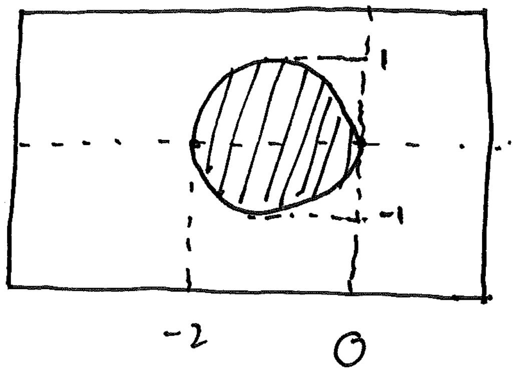
$$|\omega^i_{n+1}| \le |\omega^i_n| \,, \quad \forall\ 1 \le i \le n. \qquad \text{——} \ (5\text{-}34)$$

How. for the forward Euler method, (5-34) is ensured if $\forall i$,

$$|1 + \Delta t\, \lambda_i| < 1 \quad - \ (5.35)$$

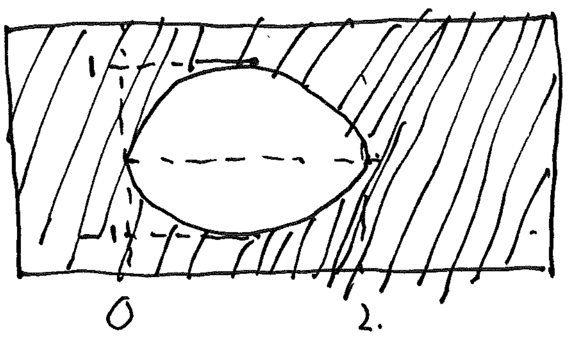
However, $\{\lambda_i\}_{i=1}^m$ are eigenvalues of A and can be complex valued. So stability regions have to be drawn in the complex plane. We provide examples below;

Forward Euler

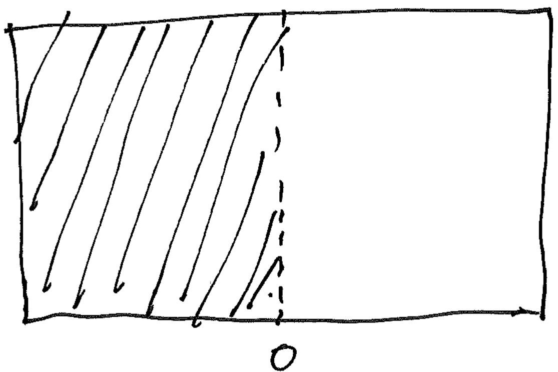


Shaded region is stability region.

Backward Euler



Trapezoidal rule



Stability regions of the RK and multi-step methods can be drawn.

5.7 Stiff problems

We find, from the discussions on stability, that Explic the Forward Euler method. Can require very small time steps, particularly if one of the eigenvalues of the . System is very large (from 5.35). However, the Backward Euler and Trapetoidal rule will be stable, even for large time steps.

Do these problems, involving very large (negative) eigenvalue occur?

The simplest example is the model ODE (5.24) with large a whose we have seen the difference between Forward and Backward Euler methods.

~~An ev~~ A more practical example. is provided by the chemical kinetics model of Chapter 1, modeling reactions

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

and given by the system:

$$u' = Au \qquad u = [u_1, u_2, u_3]$$

$$A = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix}$$

Note that the eigenvalues are given by $0, k_1, k_2$.

In many problems in Chemistry, $k_1 \gg k_2$.

For instance $k_1 = 10^6$, $k_2 = 1$,

In this case, the forward Euler requires very small time-steps, of $O(10^{-6})$ whereas the Backward Euler method works even for large $O(1)$ time steps.

Such a problem is a classic example of a stiff problem and implicit methods like Backward Euler are well suited for them as their stability region include the ~~[illegible]~~ negative (real) half of the complex plane !!!

Example 2: We consider (5.24) with the ~~Backward Euler~~ and Trapezoidal rule methods;

5.8 : BDF methods

§ Higher-order versions of the Backward Euler method, are provided by the Backward difference formula (BDF) methods of general form,

$$\alpha_0 U_n' + \alpha_1 U_{n+1} + \cdots + \alpha_\gamma U_{n+\gamma} = \Delta t \, \beta_\gamma F(U_{n+\gamma}).$$

— (5-37).

Note that $F(U) = U'$

Thus, (5-37) computes an approximation of the time-derivative at the time level $t^{n+\gamma}$ using values at γ-previous (backward) time levels, $t^{n+\gamma-1}, t^{n+\gamma-2}, \ldots, t^{n+1}, t^n$. Hence, the name BDF formulas.

Clearly, Backward-Euler method is a BDF-1 method. Other examples are

<u>BDF-2</u> $\quad 3U_{n+2} - 4U_{n+1} + U_n = 2\Delta t\, f(U_{n+2})$

<u>BDF-3</u> $\quad 11\,U_{n+3} - 18\,U_{n+2} + 9U_{n+1} - 2U_n = 6\Delta t\, f(U_{n+3})$

BDF-methods are ideally suited for stiff problems, particularly in chemistry.