Initial Value Problems: ODEs

Adaptive Runge-Kutta Schemes & Linear Multistep Methods

M.Sc. in Mathematical Modelling & Scientific Computing,
Practical Numerical Analysis

Michaelmas Term 2024, Lecture 7

Adaptive Runge-Kutta Schemes

Explicit Runge-Kutta Schemes

We continue to study the scalar first order initial value problem: find $u(t)$ such that

$$\begin{aligned} u'(t) &= f(t,u)\,, \quad t > 0\,, \\ u(0) &= u_0\,. \end{aligned} \tag{1}$$

Last time we looked at explicit Runge-Kutta schemes of the form

$$\frac{U_{n+1} - U_n}{\Delta t} = \sum_{i=1}^{s} b_i k_i$$

where

$$k_1 = f(t_n, U_n)$$

and

$$k_i = f(t_n + c_i \Delta t, U_n + \Delta t \sum_{j=1}^{i-1} a_{i,j} k_j)\,,$$

for $i = 2, \ldots, s$.

Runge-Kutta Schemes — Butcher Tableaux

The coefficients of explicit Runge-Kutta schemes are often stored as Butcher tableaux in the form

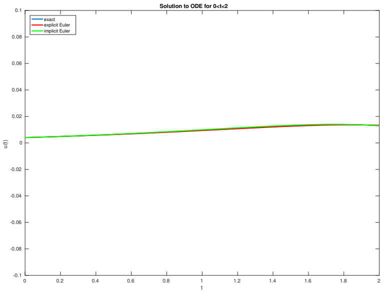
$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{2,1} & & & & \\
c_3 & a_{3,1} & a_{3,2} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s,1} & a_{s,2} & \dots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \dots & b_{s-1} & b_s
\end{array}$$

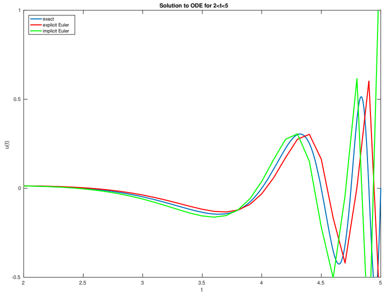
We also know that the maximum order of an s-stage Runge Kutta scheme is $p = s$ and that it is possible to construct schemes of this maximal order.

Adaptivity — Motivation

If an IVP has a solution with different timescales (i.e. a region of rapid change and a region of much less rapid change) then using a uniform timestep can be either inaccurate or inefficient.

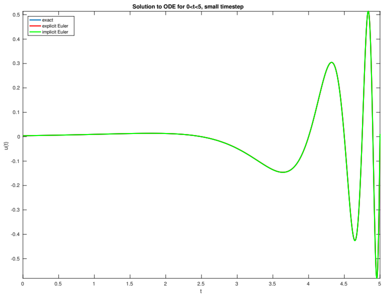
If the timestep is too large it may capture the slowly varying part of the solution but not that which is rapidly varying.





Adaptivity — Motivation

If the timestep is small it may be very inefficient for the slowly changing part of the solution.



Remedy: use a large timestep when the solution does not change rapidly and a small timestep when it does.

Runge Kutta Methods and Adaptivity (1)

One step methods can easily be modified to have adaptivity of the timestep length Δt as the function u varies. As an example, consider the fourth order Runge-Kutta method (RK4).

The background theory is that

$$\begin{aligned} \text{Trunction error} \qquad T_n &\sim K_1(\Delta t)^4 u^{(v)} \\ \text{Local error} \qquad |e_n| &\sim K_2 \Delta t |T_n| \\ \text{so} \qquad |e_n| &\sim K_3(\Delta t)^5 u^{(v)}. \end{aligned}$$

Runge Kutta Methods and Adaptivity (1)

Suppose we are at t_n and want to use a step Δt_n

1. apply RK4 over step Δt_n to get value U_a where

$$U_a \quad = \quad u_{n+1} + K_3(\Delta t_n)^5 u^{(v)} + \mathcal{O}(\Delta t_n^6)$$

2. apply RK4 *twice* over step $\Delta t_n/2$ to get value U_b

$$U_b \quad = \quad u_{n+1} + 2K_3 \left(\frac{\Delta t_n}{2}\right)^5 u^{(v)} + \mathcal{O}(\Delta t_n^6)$$

Then

$$U_a - U_b \quad \sim \quad \frac{15}{16} K_3 u^{(v)} (\Delta t_n)^5 \ \sim \ K_4 e_n.$$

Runge Kutta Methods and Adaptivity (1)

Hence for fixed Δt_n:

- if $|U_a - U_b|$ is large then so is the error $|U_a - u_{n+1}|$ so the step length should be decreased
- if $|U_a - U_b|$ is small, so is the error so we could take a larger step.

Of course we have to do more work each step since we effectively apply RK4 *three* times per timestep. The hope is that being able to use larger timesteps compensates for this.

Runge Kutta Methods and Adaptivity (1)

Since we have

$$U_a - U_b \;\sim\; \frac{15}{16} K_3 u^{(v)} (\Delta t_n)^5 \;\sim\; K_4 e_n,$$

we may write

$$|U_a - U_b| \;=\; c(\Delta t_n)^5.$$

Hence, if we require

$$|U_a - U_b| \;\le\; \mathtt{TOL}$$

then we should choose the new timestep, Δt, to satisfy

$$c(\Delta t)^5 \;=\; \frac{|U_a - U_b|}{(\Delta t_n)^5} (\Delta t)^5 \;\le\; \mathtt{TOL}$$

or equivalently

$$\Delta t \;\le\; \left(\frac{\mathtt{TOL}}{|U_a - U_b|} \right)^{1/5} \Delta t_n.$$

Runge Kutta Methods and Adaptivity (1)

Algorithm: User provides start time, end time, tolerance.

```
1. Set TOL=tolerance
2. Set $t_0$ =start time
3. while $t_n \leq$ end time
    3.1 apply RK4 to determine $U_a$ and $U_b$ with current $\Delta t_n$
    3.2 if $|U_a - U_b| >$TOL, step fails, set

            $\Delta t_n = \left( \frac{\text{TOL}}{|U_a - U_b|} \right)^{1/5} \Delta t_n$

        and go back to 3.1. (This reduces the step and repeats.)
        else $|U_a - U_b| \leq$TOL, set

            $\Delta t_{n+1} = \left( \frac{\text{TOL}}{|U_a - U_b|} \right)^{1/5} \Delta t_n$    (2)
            $U_{n+1} = U_b$
            $t_{n+1} = t_n + \Delta t_n$
            $n = n + 1$

        (this increases the step length for *next* step).
    end while
```

Runge Kutta Methods and Adaptivity (1)

As we have been dealing with local error the lengthening of step can be misleading, the global error has order $(\Delta t)^4$ so in (2) above can use

$$\Delta t_{n+1} \quad = \quad \left(\frac{\mathtt{TOL}}{|U_a - U_b|} \right)^{1/4} \Delta t_n.$$

This is more robust in practice.

Runge Kutta Methods and Adaptivity (2)

An alternative to the method described above is to use two Runge Kutta methods, one of order p and one of order $\tilde{p} \geq p+1$.

Let U_{n+1} be the numerical approximation to $u(t_{n+1})$ using the pth order method and let $\tilde{U}_{n+1}$ be the numerical approximation to $u(t_{n+1})$ using the $\tilde{p}$th order method.

Then (making error free assumption, i.e. all earlier iterates are exact)

$$\begin{aligned} U_{n+1} &= u(t_{n+1}) + c\Delta t^{p+1} + \mathcal{O}(\Delta t^{p+2}) \,, &(3)\\ \tilde{U}_{n+1} &= u(t_{n+1}) + \mathcal{O}(\Delta t^{p+2}) \,, &(4)\end{aligned}$$

as $\Delta t \to 0$. Here c depends on the derivative of u. Subtracting gives

$$U_{n+1} - \tilde{U}_{n+1} \approx c\Delta t^{p+1} \,,$$

and substituting this in (3) gives

$$U_{n+1} - u(t_{n+1}) \approx U_{n+1} - \tilde{U}_{n+1} \,.$$

Runge Kutta Methods and Adaptivity (2)

We can use this final equation

$$U_{n+1} - u(t_{n+1}) \;\approx\; U_{n+1} - \tilde{U}_{n+1} \,,$$

to determine when to refine the mesh in an adaptive algorithm.

Now the idea is to choose the ERK schemes so that the pth order method has nodes and a matrix which are a subset of those in the $\tilde{p}$th order method so that the values can be re-used. This approach is called an *embedded RK pair*.

Runge Kutta Methods and Adaptivity (2)

Example 1: Choose the pair consisting of the Butcher tableaux

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

and

$$\begin{array}{c|ccc} 0 & & & \\ \frac{2}{3} & \frac{2}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{8} & \frac{3}{8} \end{array}$$

Runge Kutta Methods and Adaptivity (2)

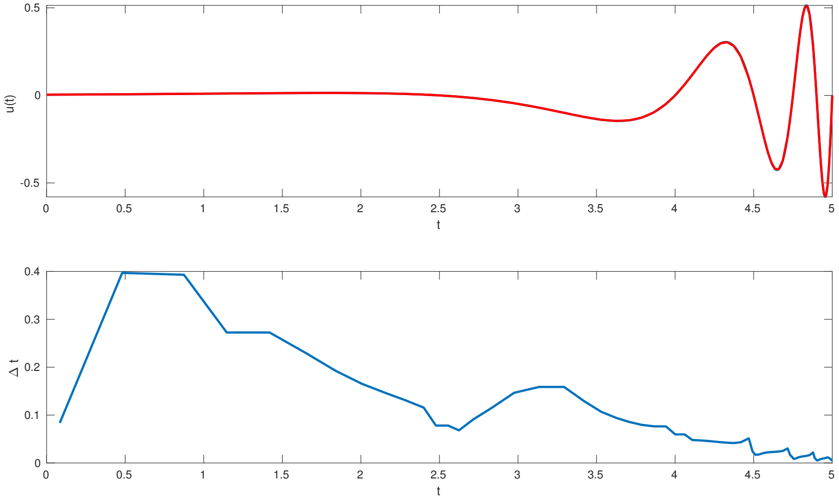
Example 2: Choose the pair consisting of the Butcher tableaux

$$
\begin{array}{c|ccccc}
0 & & & & & \\
1/4 & 1/4 & & & & \\
3/8 & 3/32 & 9/32 & & & \\
12/13 & 1932/2197 & -7200/2197 & 7296/2197 & & \\
1 & 439/216 & -8 & 3680/513 & -845/4104 & \\
\hline
 & 25/216 & 0 & 1408/2565 & 2197/4104 & -1/5
\end{array}
$$

and

$$
\begin{array}{c|cccccc}
0 & & & & & & \\
1/4 & 1/4 & & & & & \\
3/8 & 3/32 & 9/32 & & & & \\
12/13 & 1932/2197 & -7200/2197 & 7296/2197 & & & \\
1 & 439/216 & -8 & 3680/513 & -845/4104 & & \\
1/2 & -8/27 & 2 & -3544/2565 & 1859/4104 & -11/40 & \\
\hline
 & 16/135 & 0 & 6656/12825 & 28561/56430 & -9/50 & 2/55
\end{array}
$$

Results



Linear Multistep Methods

Linear Multistep Methods: General Form

We continue to study the scalar first order initial value problem: find $u(t)$ such that

$$\begin{aligned} u'(t) &= f(t,u)\,, \quad t>0\,, \\ u(0) &= u_0\,. \end{aligned} \tag{5}$$

The general form of a linear k-step method for the solution of (5) is

$$\sum_{j=0}^{k} \alpha_j U_{n+j} = \Delta t \sum_{j=0}^{k} \beta_j f(t_{n+j}, U_{n+j})\,,$$

where $\{\alpha\}_{j=0}^{k}$ and $\{\beta\}_{j=0}^{k}$ are real coefficients with $\alpha_k \neq 0$ and $|\alpha_0| + |\beta_0| \neq 0$.

Note that if $\beta_k = 0$ the method is explicit, otherwise it is implicit.

Linear Multistep Methods: Example

If we integrate (5) over the interval (t_n, t_{n+2}) we have

$$u(t_{n+2}) - u(t_n) = \int_{t_n}^{t_{n+2}} f(t, u(t))\mathrm{d}t \; .$$

We can now approximate the integral using Simpson's rule to give

$$\begin{aligned} u(t_{n+2}) - u(t_n) \;\; \approx \;\; & \frac{\Delta t}{3}\left(f(t_n, u(t_n)) + 4f(t_{n+1}, u(t_{n+1}))\right. \\ & \left. + f(t_{n+2}, u(t_{n+2}))\right) \; . \end{aligned}$$

Thus an example of a linear 2-step method is

$$U_{n+2} - U_n = \frac{\Delta t}{3}\left(f_n + 4f_{n+1} + f_{n+2}\right) \; ,$$

where $f_n = f(t_n, U_n)$.

Linear Multistep Methods: First and Second Characteristic Polynomials

In the same way Runge-Kutta methods are summarised with Butcher tables, linear multistep methods can be summarised with two polynomials

$$\begin{aligned} \rho(z) &= \sum_{j=0}^{k} \alpha_j z^j \ , \\ \sigma(z) &= \sum_{j=0}^{k} \beta_j z^j \ , \end{aligned}$$

known as the first and second characteristic polynomials, respectively.

Linear Multistep Methods: Truncation Error

The truncation error is defined as

$$T_n = \frac{\sum_{j=0}^{k} \alpha_j u(t_{n+j}) - \Delta t \sum_{j=0}^{k} \beta_j f(t_{n+j}, u(t_{n+j}))}{\Delta t \sum_{j=0}^{k} \beta_j}$$

where $\sum_{j=0}^{k} \beta_j = \sigma(1) \neq 0$.

Taylor series expansions allow us to re-write this as

$$T_n = \frac{\sum_{s=0}^{\infty} C_s \Delta t^s u^{(s)}(t_n)}{\Delta t \sigma(1)} ,$$

where

$$C_0 = \sum_{j=0}^{k} \alpha_j , \quad C_1 = \sum_{j=1}^{k} j\alpha_j - \sum_{j=0}^{k} \beta_j ,$$

$$C_s = \sum_{j=1}^{k} \frac{j^s}{s!}\alpha_j - \sum_{j=1}^{k} \frac{j^{s-1}}{(s-1)!}\beta_j , \quad s \geq 2 .$$

Linear Multistep Methods: Truncation Error Example

Consider the scheme

$$U_{n+2} - U_n \;=\; \frac{\Delta t}{3}\left(f_{n+2} + 4f_{n+1} + f_n\right) \;.$$

We have $\sigma(1) = 2 \neq 0$ and

$$\begin{aligned}
C_0 &= \sum_{j=0}^{k} \alpha_j \;=\; 1 - 1 \;=\; 0\,, \\
C_1 &= \sum_{j=1}^{k} j\alpha_j - \sum_{j=0}^{k} \beta_j \;=\; 2 - \frac{1}{3}(1+4+1) \;=\; 0\,, \\
&\;\vdots \\
C_4 &= \sum_{j=1}^{k} \frac{j^4}{4!}\alpha_j - \sum_{j=1}^{k} \frac{j^3}{3!}\beta_j \;=\; \frac{16}{24} - \frac{1}{3}\frac{1}{6}(8+4) \;=\; 0\,, \\
C_5 &= \sum_{j=1}^{k} \frac{j^5}{5!}\alpha_j - \sum_{j=1}^{k} \frac{j^4}{4!}\beta_j \;=\; \frac{32}{120} - \frac{1}{3}\frac{1}{24}(16+4) \;\neq\; 0\,.
\end{aligned}$$

Linear Multistep Methods: Truncation Error Example

Thus for this scheme we have

$$T_n = \frac{C_5}{2}\Delta t^4 u^{(5)}(t_n) + \mathcal{O}(\Delta t^5) \; .$$

Linear Multistep Methods: Order

Recall that the order of a method is p if p is the largest integer such that $T_n = \mathcal{O}(\Delta t^p)$.

Hence a linear k-step method is of order p if $C_0 = C_1 = \ldots C_p = 0$ and $C_{p+1} \neq 0$.

The scheme based on Simpson's rule is 4th order.

Linear Multistep Methods: Stability

Recall that for Runge-Kutta schemes we defined the interval of absolute stability as the set of values of $\lambda\Delta t$ for which the numerical solution U_n generated by applying the scheme to the problem

$$\begin{aligned} u'(t) &= \lambda u\,, \quad t > 0\,, \\ u(0) &= 1\,. \end{aligned}$$

with $\lambda < 0$ satisfies $\lim_{n\to\infty} U_n = 0$.

The same ideas hold for linear multistep methods. Here the numerical solution satisfies

$$\sum_{j=0}^{k} \alpha_j U_{n+j} = \lambda\Delta t \sum_{j=0}^{k} \beta_j U_{n+j}\,.$$

Linear Multistep Methods: Stability

Thus the numerical solution is

$$U_n \;=\; \sum_{r=1}^{q} p_r(n) z_r^n \,,$$

where the z_r are the *distinct* roots of the polynomial

$$\sum_{j=0}^{k} (\alpha_j - \lambda \Delta t \beta_j) z^j \;=\; \rho(z) - \lambda \Delta t \sigma(z) \;=\; 0 \,,$$

and the $p_r(n)$ are polynomials in n of degree one less than the multiplicity of the root z_r.

This means that

$$\lim_{n \to \infty} U_n \;=\; 0 \,,$$

if all the roots of the stability polynomial $\rho(z) - \lambda \Delta t \sigma(z)$ satisfy $|z| < 1$.

Linear Multistep Methods: Stability

Again consider the scheme

$$U_{n+2} - U_n = \frac{\Delta t}{3}\left(f_{n+2} + 4f_{n+1} + f_n\right) .$$

We have

$$\begin{aligned}\rho(z) &= z^2 - 1 , \\ \sigma(z) &= \frac{1}{3}(z^2 + 4z + 1) ,\end{aligned}$$

and the stability polynomial is

$$z^2 - 1 - \frac{\lambda \Delta t}{3}(z^2 + 4z + 1) = 0 .$$

The roots of the stability polynomial are

$$z_{\pm} = \frac{2\lambda\Delta t \pm \sqrt{5(\lambda\Delta t)^2 - 9}}{3 - \lambda\Delta t} .$$

It can be shown that $|z_{\pm}| < 1$ if $\lambda\Delta t \in (-3/2, 0)$.

Linear Multistep Methods: Initial Conditions

The general form of a linear k-step method is

$$\sum_{j=0}^{k} \alpha_j U_{n+j} \;=\; \Delta t \sum_{j=0}^{k} \beta_j f(t_{n+j}, U_{n+j}) \; .$$

In order to compute U_{n+k} we need to know U_n, U_{n+1}, $\ldots$, U_{n+k-1}.

In particular, in order to compute U_k we need k initial conditions: U_0, U_1, $\ldots$, U_{k-1}. We use $U_0 = u_0$ but we need a consistent way to approximate the remaining conditions.

Numerical initial conditions are called *consistent* with the initial condition $u(0) = u_0$ if they are of the form

$$U_i \;=\; \eta_i(\Delta t)$$

where

$$\lim_{\Delta t \to 0} \eta_i(\Delta t) \;=\; u_0 \; ,$$

for $i = 0, 1, \ldots, k-1$.

Linear Multistep Methods: Zero Stability

A linear k-step method for the ODE (5) is said to be zero-stable if there exists a constant K such that, for any two sequences (U_n) and $(\hat{U}_n)$, which have been generated by the same formulae but with different initial data U_0, U_1, ..., U_{k-1} and $\hat{U}_0$, $\hat{U}_1$, ..., $\hat{U}_{k-1}$, respectively, we have

$$|U_n - \hat{U}_n| \quad \leq \quad K \max\left\{|U_0 - \hat{U}_0|, |U_1 - \hat{U}_1|, \ldots, |U_{k-1} - \hat{U}_{k-1}|\right\}$$

for $t_n \leq T_{\mathrm{final}}$, and as Δt tends to 0.

It can be shown that a linear multi-step method is zero-stable if, and only if, its first characteristic polynomial has zeros inside the closed unit disc, with any which lie on the unit circle being simple. This is often known as the root condition.

Linear Multistep Methods: Zero Stability Example

The scheme

$$U_{n+2} - U_n = \frac{\Delta t}{3} \left(f_{n+2} + 4 f_{n+1} + f_n \right) ,$$

has first characteristic polynomial

$$\rho(z) = z^2 - 1 .$$

The roots of $\rho(z)$ are $z = \pm 1$ which lie on the unit circle and are simple. Thus the scheme is zero stable.

Linear Multistep Methods: Dahlquist's Theorem

Dahlquist's Theorem states:

For a linear multi-step method that is consistent with the ordinary differential equation (5) where f is assumed to satisfy a Lipschitz condition, and starting with consistent initial data, zero-stability is necessary and sufficient for convergence.

Moreover if the solution $u(t)$ has continuous derivative of order $p+1$ and consistency error $\mathcal{O}(\Delta t^p)$, then the global error $e_n = u(t_n) - U_n$ is also $\mathcal{O}(\Delta t^p)$.