MATH3511/6111: Scientific Computing

14. Ordinary Differential Equations

Lindon Roberts

Semester 1, 2022

Office: Hanna Neumann Building #145, Room 4.87
Email: lindon.roberts@anu.edu.au
Based on lecture notes written by S. Roberts, L. Stals, Q. Jin, M. Hegland, K. Duru, A. Paganini.

Australian National University

Ordinary Differential Equation

An ordinary differential equation (ODE) is an equation in terms of the derivatives of a function $u(t)$.

The order is the highest derivative present in the equation (e.g. first-order ODE only uses $u(t)$ and $u'(t)$).

A common form is the explicit form of a first-order ODE:

$$\frac{du}{dt} = f(t, u(t)),$$

where $f(t, u)$ is some given function. To ensure we have a unique solution, we need to impose some extra condition.

Usually this is a starting value for u:

$$u(0) = u_0.$$

In this case, the ODE is an initial value problem (IVP).

Ordinary Differential Equations: Types

Higher-order ODEs such as

$$\frac{d^2u}{dt^2} - \frac{du}{dt} = f(t, u(t)),$$

generally require more than one extra condition. For a second-order ODE, this could be two initial conditions (still an IVP):

$$u(0) = u_0, \qquad \text{and} \qquad u'(0) = v_0.$$

Alternatively we could prescribe $u(t)$ at two different locations:

$$u(0) = u_0, \qquad \text{and} \qquad u(1) = u_1.$$

These are called boundary value problems (BVPs).

We will focus on IVPs, as BVPs are much more complicated.

Ordinary Differential Equations: Types

A linear ODE is one which defined as a polynomial in $u(t)$ and its derivatives:

$$a_0(t)u(t) + a_1(t)u'(t) + a_2(t)u''(t) + \cdots + a_n(t)u^{(n)}(t) = b(t),$$

for given functions $a_i(t)$ and $b(t)$.

Many common ODEs are linear, but there are also many important nonlinear ODEs. Examples of nonlinear ODEs include

$$u''(t) = u^2, \qquad \text{or} \qquad (u'(t))^2 + \sin(u(t)) = t.$$

If there is no explicit dependence on t, the ODE is autonomous (i.e. t only appears inside $u(t)$, $u'(t)$, etc.). For example, the first ODE above is autonomous, the second is not; the linear ODE above is not autonomous.

Systems of ODEs

Another common situation is to look at a system of ODEs, where we need to find multiple functions. For example, $\boldsymbol{u}'(t) = A(t)\boldsymbol{u}(t)$ for some matrix $A(t)$ (changing with t), is a first-order linear system of ODEs.

An important trick

Higher-order ODEs can be written as first-order systems of ODEs.

Define the variables to be $\boldsymbol{u}(t) = (u(t), u'(t), \ldots, u^{(n-1)}(t))$ and add the extra equations $\frac{du'(t)}{dt} = u''(t)$, etc. For example, the second-order nonlinear ODE

$$au''(t) + bu'(t) - u^2 = t^3,$$

is equivalent to setting $v(t) = u'(t)$ and rearranging to

$$\frac{d}{dt}\begin{bmatrix} u(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \frac{1}{a}\left(t^3 - bv(t) + u(t)^2\right) \end{bmatrix}.$$

Because of this, we will focus on solving first-order ODEs.

Autonomous Form

Actually, a similar trick allows us to turn every non-autonomous ODE into an autonomous ODE.

We do this by adding an extra variable defined by $u_0(t) = t$, and so $u_0'(t) = 1$ and $u_0(0) = 0$.

For example: we previously had the second-order ODE (or first-order system)

$$au''(t) + bu'(t) - u^2 = t^3, \qquad \text{or} \qquad \frac{d}{dt}\begin{bmatrix} u(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \frac{1}{a}\left(t^3 - bv(t) + u(t)^2\right) \end{bmatrix},$$

where $v(t) = u'(t)$.

Now, add an extra variable again: defining $w(t) = t$ we rearrange to get the autonomous ODE

$$\frac{d}{dt}\begin{bmatrix} u(t) \\ v(t) \\ w(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \frac{1}{a}\left(w(t)^3 - bv(t) + u(t)^2\right) \\ 1 \end{bmatrix},$$

with initial conditions $u(0) = u_0$, $v(0) = v_0$, $w(0) = 0$.

ODE Examples: Growth and Decay

ODEs are a fundamental feature in mathematical models across many disciplines. We will give a few examples.

Exponential growth:

$$\boxed{u'(t) = \alpha u, \qquad \text{with} \qquad u(0) = u_0}$$

where α is the growth rate. The solution is $u(t) = u_0 e^{\alpha t}$.

Growth and decay:

$$\boxed{u'(t) = \alpha - \beta u, \qquad \text{with} \qquad u(0) = u_0}$$

The parameter α is a growth rate and β is a decay rate.

The ODE is first-order, linear and autonomous. The solution is

$$u(t) = e^{-\beta t} u_0 + (1 - e^{-\beta t}) \frac{\alpha}{\beta}.$$

ODE Examples: Growth and Decay

Growth and decay:

$$u'(t) = \alpha - \beta u, \qquad \text{with} \qquad u(0) = u_0$$

Autonomous ODEs can have 'stationary solutions', $u(t) = \text{const}$. Here, if $u_0 = \alpha/\beta$ then $u(t) = u_0$ for all t is the solution.

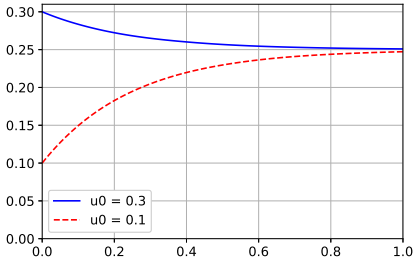

Figure 1. Solutions $u(t)$ for $\alpha = 1$, $\beta = 4$ and different choices of u_0.

ODE Examples: Mechanics

Mechanics: A particle is moving in the presence of drag and gravity. Newton's second law ($F = ma$) gives an ODE like

$$\underbrace{mu''(t)}_{\text{Force}} = \underbrace{-\beta(u'(t))^2}_{\text{drag}} - \underbrace{\frac{\gamma}{u(t)^2}}_{\text{gravity}}.$$

This is a second-order, nonlinear, autonomous ODE.

Like before, we can convert it to a first-order system of ODEs by defining $v(t) = mu'(t)$ (momentum). Then

$$\frac{d}{dt}\begin{bmatrix} u(t) \\ v(t) \end{bmatrix} = f\left(\begin{bmatrix} u(t) \\ v(t) \end{bmatrix}\right), \qquad \text{where} \qquad f\left(\begin{bmatrix} u \\ v \end{bmatrix}\right) = \begin{bmatrix} v/m \\ -\beta(v/m)^2 - \gamma/u^2 \end{bmatrix}.$$

For this to be an IVP, we need to specify the starting position $u(0) = u_0$ and momentum $v(0) = v_0$.

ODE Examples: Chemical Reactions

Chemical Reactions: Burn hydrogen and oxygen to get water: $2H + O \to H_2O$.

The law of mass action says that the rate of reaction is proportional to the concentrations of the inputs:

$$\text{Reaction rate} = \kappa[H]^2[O],$$

where $[X]$ is the concentration of substance X. Therefore the concentrations of each substance follow the ODEs

$$\frac{d[H]}{dt} = -2\kappa[H]^2[O], \qquad \frac{d[O]}{dt} = -\kappa[H]^2[O], \qquad \frac{d[H_2O]}{dt} = +\kappa[H]^2[O].$$

With suitable initial conditions, this gives a first-order, nonlinear, autonomous system of ODEs:

$$\frac{d}{dt}\begin{bmatrix} c_H(t) \\ c_O(t) \\ c_{H_2O}(t) \end{bmatrix} = \kappa c_H(t)^2 c_O(t) \begin{bmatrix} -2 \\ -1 \\ 1 \end{bmatrix}.$$

ODE Examples: Epidemiology

Epidemiology: SIR model for diseases. A person is either **s**usceptible, **i**nfected or **r**emoved (immune or dead). The number of people changing between each category is proportional to the population in each category:

$$\frac{d}{dt}\begin{bmatrix} S(t) \\ I(t) \\ R(t) \end{bmatrix} = f\left(\begin{bmatrix} S(t) \\ I(t) \\ R(t) \end{bmatrix}\right), \qquad \text{where} \qquad f\left(\begin{bmatrix} S \\ I \\ R \end{bmatrix}\right) = \begin{bmatrix} -\alpha SI \\ \alpha SI - \beta I \\ \beta I \end{bmatrix}.$$

Important property: conservation (of # people).

$$\frac{d}{dt}(S(t) + I(t) + R(t)) = (-\alpha SI) + (\alpha SI - \beta I) + (\beta I) = 0.$$

Many ODEs have important conservation properties (e.g. conservation of energy, mass, ...).

ODE Examples: Epidemiology

Example SIR model: $\alpha = 7$, $\beta = 2$, starting with 99% susceptible, 1% infected.

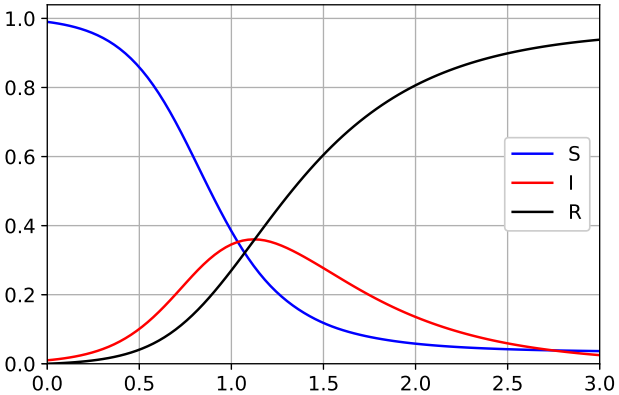

Figure 2. SIR model example.

Existence and Uniqueness

Before we start figuring out how to solve ODEs numerically, we should first check that solutions exist!

Theorem (Picard-Lindelöf; Émile Picard, Ernst Lindelöf, late 1800s)

Suppose there is some open set $\Omega \in \mathbb{R}^2$ containing (t_0, u_0) inside which $f(t,u)$ is continuous in t and uniformly Lipschitz continuous in u: that is, there exists $M > 0$ such that

$$|f(t,u_2) - f(t,u_1)| \leq M|u_2 - u_1|, \qquad \forall (t,u_1),(t,u_2) \in \Omega.$$

Then the ODE

$$u'(t) = f(t,u), \qquad \text{with} \qquad u(t_0) = u_0,$$

has a unique (and differentiable!) solution $u(t)$ in some interval around t_0, $[t_0 - \epsilon, t_0 + \epsilon]$.

A similar theorem holds for first-order systems of ODEs (initial value problems).

Existence and Uniqueness

The different conditions on $f(t,u)$ are actually needed! For example,

$$u'(t) = 3(u-1)^{2/3}, \qquad \text{with} \qquad u(0) = 1,$$

has two solutions $u(t) = 1 + t^3$ and $u(t) = 1$. Here, $f(t,u)$ is not Lipschitz continuous in u.

Alternatively,

$$u'(t) = 1 + u(t)^2, \qquad \text{with} \qquad u(0) = 0,$$

has a unique solution $u(t) = \tan(t)$, but only for $t \in (-\pi/2, \pi/2)$ (blows up at $t = \pm\pi/2$ so $u'(t)$ doesn't exist). This shows we may not be able to define a solution for all t.

Solving ODEs: Basic Idea

From the Fundamental Theorem of Calculus (also in proof of Picard-Lindelöf), the solution to

$$u'(t) = f(t, u), \qquad \text{with} \qquad u(t_0) = u_0,$$

is

$$u(t) = u_0 + \int_{t_0}^{t} f(s, u(s))ds.$$

So, we basically need to find ways to approximate this integral!

Perhaps unsurprisingly, the basic technique we will use to derive algorithms for solving ODEs is quadrature (numerical integration).

Is this harder than quadrature? Yes, because u is calculated by integrating $f(t, u)$, which itself depends on u!

General Framework

$$u'(t) = f(t,u), \quad u(t_0) = u_0 \qquad \Longleftrightarrow \qquad u(t) = u_0 + \int_{t_0}^{t} f(s,u(s))ds$$

In general, we want to find $u(t)$ for some range of t, say $[t_0, T]$.

Just like for quadrature, we split the interval $[t_0, T]$ into sub-intervals, and use a composite rule by summing the rule in each segment.

Let's take our grid to be equally-spaced: $t_k = t_0 + kh$ for $k = 0, \ldots, n$, where $h = (T - t_0)/n$ (so $t_n = T$). Define u_k to be our approximation of $u(t_k)$.

Then we have u_0 given already (initial condition), and we want to estimate

$$u_{k+1} \approx u_k + \int_{t_k}^{t_{k+1}} f(s,u(s))ds.$$

The error in any ODE scheme is $e_k = |u_k - u(t_k)|$.

Explicit Euler Method

$$\boxed{u_{k+1} \approx u_k + \int_{t_k}^{t_{k+1}} f(s, u(s))ds}$$

An obvious choice to approximate the integral is the left Riemann sum

$$\int_{t_k}^{t_{k+1}} f(s, u(s))ds \approx (t_{k+1} - t_k)f(t_k, u_k).$$

Note there are two sources of error: approximating the integral with a left Riemann sum, and approximating $u(t_k)$ with u_k.

This approximation gives us the explicit Euler scheme (Leonhard Euler, 1768):

$$u_{k+1} = u_k + (t_{k+1} - t_k)f(t_k, u_k) = u_k + hf(t_k, u_k).$$

This is an explicit formula for u_{k+1} (since all RHS quantities are known at step k). It is a one-step method because we only need the values at one previous time step.

Explicit Euler Method

$$\boxed{u_{k+1} = u_k + hf(t_k, u_k)}$$

What is the error in the explicit Euler scheme? We need to quantify the two sources of error:

- Approximating the integral with a left Riemann sum
- Approximating $u(t_k)$ with u_k.

We measure the first error using the local discretisation/truncation error

$$L(t,h) := \frac{u(t+h) - u(t)}{h} - f(t, u(t)). \qquad \leftarrow \text{(for explicit Euler only)}$$

Using this, we see that the exact solution $u(t)$ almost satisfies the explicit Euler scheme

$$u(t_{k+1}) = u(t_k) + hf(t_k, u(t_k)) + hL(t_k, h).$$

Explicit Euler Method

For explicit Euler the local truncation error is (using Taylor series):

$$\begin{aligned} L(t,h) &= \frac{u(t+h)-u(t)}{h} - f(t,u(t)), \\ &= \frac{\left[u(t)+u'(t)h+\mathcal{O}(h^2)\right]-u(t)}{h} - f(t,u(t)), \\ &= \frac{\left[u(t)+f(t,u(t))h+\mathcal{O}(h^2)\right]-u(t)}{h} - f(t,u(t)), \\ &= \mathcal{O}(h). \end{aligned}$$

We can then calculate the overall error:

$$\begin{aligned} u(t_{k+1}) - u_{k+1} &= \left[u(t_k) + hf(t_k,u(t_k)) + hL(t_k,h)\right] - \left[u_k + hf(t_k,u_k)\right], \\ &= \left[u(t_k)-u_k\right] + h\left[f(t_k,u(t_k)) - f(t_k,u_k)\right] + hL(t_k,h). \end{aligned}$$

i.e. New error is (old error) plus (error from wrong u in integrand $f(t,u)$) plus (quadrature error = local truncation error).

Explicit Euler Method

Taking absolute values, the error $e_k = |u_k - u(t_k)|$ for explicit Euler satisfies

$$\boxed{e_{k+1} \leq e_k + h|f(t_k, u(t_k)) - f(t_k, u_k)| + h|L(t_k, h)|}$$

If $f(t, u)$ is uniformly Lipschitz continuous in u (the assumption from Picard-Lindelöf), then

$$|f(t_k, u(t_k)) - f(t_k, u_k)| \leq M|u(t_k) - u_k| = Me_k.$$

Using the abbreviation $L_k = |L(t_k, h)|$, this gives:

$$e_{k+1} \leq (1 + hM)e_k + hL_k.$$

Question

Our errors are growing over time. How bad are they at the final time step e_n (at time $T = nh$)?

Useful Lemma

Lemma

If a sequence $d_0, d_1, d_2, \ldots$ satisfies $d_{j+1} \leq Cd_j + D$ for all $j = 0, 1, \ldots$, then

$$d_n \leq C^n d_0 + D\frac{C^n - 1}{C - 1}, \qquad \text{for all } n = 0, 1, 2, \ldots.$$

Proof: By induction. First check $n = 0$:

$$d_0 = C^0 d_0 + D\frac{C^0 - 1}{C - 1}.$$

Assume true for n and prove true for $n + 1$:

$$\begin{aligned} d_{n+1} \leq Cd_n + D \leq C\left(C^n d_0 + D\frac{C^n - 1}{C - 1}\right) + D &= C^{n+1}d_0 + D\left(\frac{C(C^n - 1)}{C - 1} + 1\right), \\ &= C^{n+1}d_0 + D\left(\frac{C^{n+1} - C + C - 1}{C - 1}\right) = C^{n+1}d_0 + D\frac{C^{n+1} - 1}{C - 1}. \end{aligned}$$

Explicit Euler Method

$$\boxed{e_{k+1} \leq (1 + hM)e_k + hL_k}$$

Using lemma with $d_j = e_j$, $C = 1 + hM$ and $D = hL$ where $L = \max_j L_j$, we get

$$e_n \leq (1 + hM)^n e_0 + hL\frac{(1 + hM)^n - 1}{1 + hM - 1}.$$

Since $n = T/h = TM/(hM)$ and the identity $1 + x \leq \exp(x)$, we get

$$\begin{aligned} e_n &\leq (1 + hM)^{TM/(hM)} e_0 + hL\frac{(1 + hM)^{TM/(hM)} - 1}{1 + hM - 1}, \\ &\leq \exp(TM)e_0 + \frac{L(\exp(TM) - 1)}{M}. \end{aligned}$$

This is a global error bound (i.e. total error accumulated over n time steps).

Usually $e_0 = 0$ (exact initial conditions), so since $L = \mathcal{O}(h)$ we have $e_n = \mathcal{O}(h)$. Therefore the explicit Euler scheme is a first-order method.

Explicit Euler: Example

Solving $u'(t) = -1.5u$ with $u(0) = 1$ (true solution $u(t) = e^{-1.5t}$) on $[0, 2]$ with $h = 0.25$:

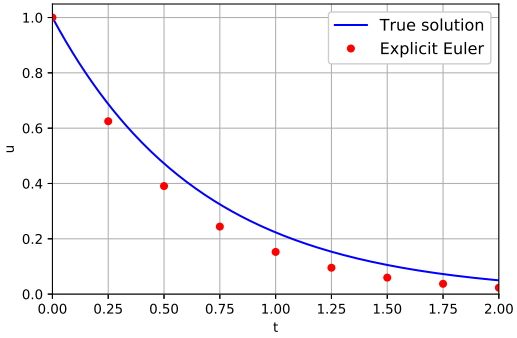

Figure 3. Explicit Euler example.

Error at $T = 2$ is approx. 0.0265. Max error at any timestep is approx. 0.0817 at $t = 0.5$.

Explicit Euler: Example

Now look at final and maximum error for decreasing values of h:

```
    h      Final error       Max error
2.0e-01  2.153954e-02     6.356966e-02
2.0e-02  2.234560e-03     5.588367e-03
2.0e-03  2.239855e-04     5.525101e-04
2.0e-04  2.240362e-05     5.518882e-05
2.0e-05  2.240412e-06     5.518261e-06
2.0e-06  2.240417e-07     5.518199e-07
```

Both errors are decreasing at a rate of $\mathcal{O}(h)$, as expected.

General One-Step Methods

The explicit Euler scheme is an example of a one-step method

$$u_{k+1} = u_k + h\phi(t_k, u_k),$$

where ϕ is some function depending only on the solution value u_k (not u_{k-1}, u_{k-2}, etc.).

Examples include:

- Explicit Euler: $\phi(t_k, u_k) = f(t_k, u_k)$
- Heun's method: $\phi(t_k, u_k) = \frac{1}{2}f(t_k, u_k) + \frac{1}{2}f(t_k + h, u_k + hf(t_k, u_k))$
- Midpoint method: $\phi(t_k, u_k) = f(t_k + h/2, u_k + hf(t_k, u_k)/2)$

These methods are called explicit because the RHS only depends on known quantities (implicit is where the RHS depends on u_{k+1}, so there is an equation to solve using rootfinding methods).

General One-Step Methods

The local truncation error for a general one-step method is

$$L(t,h) := \frac{u(t+h)-u(t)}{h} - \phi(t,u(t)).$$

The method is consistent if

$$\lim_{h\to 0^+} \sup_{t\in[0,T]} L(t,h) = 0,$$

i.e. $L(t,h) \to 0$ as $h \to 0$, uniformly in t.

The method is order p if

$$L(t,h) = \mathcal{O}(h^p),$$

as $h \to 0$, uniformly in t. That is, for h sufficiently small, $L(t,h) \leq Ch^p$ for some constant $C > 0$ independent of t.

General One-Step Methods

A general one-step method is stable if ϕ is uniformly Lipschitz continuous in u.

That is, there exists $M > 0$ such that

$$|\phi(t, u_1) - \phi(t, u_2)| \leq M|u_1 - u_2|, \qquad \forall u_1, u_2 \in \mathbb{R}, \forall t \in [0, T].$$

Theorem

If a one-step method is consistent and stable, then it is convergent (i.e. $e_n \to 0$ as $h \to 0$, provided $e_0 = 0$).

Proof: Same as for explicit Euler. By definition of $L(t, h)$, we get

$$\begin{aligned} u(t_{k+1}) - u_{k+1} &= [u(t_k) + h\phi(t_k, u(t_k)) + hL(t_k, h)] - [u_k + h\phi(t_k, u_k)], \\ &= [u(t_k) - u_k] + h[\phi(t_k, u(t_k)) - \phi(t_k, u_k)] + hL(t_k, h). \end{aligned}$$

Stability gives $|\phi(t_k, u(t_k)) - \phi(t_k, u_k)| \leq Me_k$ and so

$$e_{k+1} \leq (1 + hM)e_k + h|L(t_k, h)|.$$

Defining $L = \max_k |L(t_k, h)|$ and using the lemma like before we get

$$e_n \leq \exp(TM)e_0 + \frac{L(\exp(TM) - 1)}{M}.$$

Consistency says $L \to 0$ as $h \to 0$, so $e_n \to 0$ (provided that $e_0 = 0$).

Theorem

If a one-step method is order p and stable, then $e_n = \mathcal{O}(h^p)$ as $h \to 0$ (provided $e_0 = 0$).

Proof: From before we have (if the method is stable)

$$e_n \leq \exp(TM)e_0 + \frac{L(\exp(TM) - 1)}{M}.$$

Since the method is order p, we have $L = \mathcal{O}(h^p)$ as $h \to 0$.

Hence $e_n = \mathcal{O}(h^p)$ (provided $e_0 = 0$).

This means that the order of the local truncation error determines the order of the (global) error, provided we have stability and $e_0 = 0$.

Higher-Order Methods

Question

Explicit Euler is order 1. Are there any higher-order methods?

Yes! (probably unsurprisingly by now)

We have to use a better quadrature rule to approximate

$$u(t_{k+1}) = u(t_k) + \int_{t_k}^{t_{k+1}} f(s, u(s))ds.$$

For example, using the trapezoidal rule we get the implicit trapezoidal method

$$u_{k+1} = u_k + \frac{h}{2}[f(t_k, u_k) + f(t_{k+1}, u_{k+1})].$$

Unfortunately, this is method is implicit (since the RHS depends on u_{k+1}) — we need to find u_{k+1} using a rootfinding algorithm at every iteration. By our definition, it is not a one-step method.

Heun's Method

$$\boxed{u_{k+1} = u_k + \frac{h}{2}[f(t_k, u_k) + f(t_{k+1}, u_{k+1})]}$$

We can modify the implicit trapezoidal method by approximating u_{k+1} in the last term with explicit Euler:

$$u_{k+1} = u_k + \frac{h}{2}\left[f(t_k, u_k) + f\left(t_{k+1}, u_k + hf(t_k, u_k)\right)\right].$$

This is Heun's method (Karl Heun, 1900).

It is a general first-order method with

$$\phi(t_k, u_k) = \frac{1}{2}f(t_k, u_k) + \frac{1}{2}f(t_k + h, u_k + hf(t_k, u_k)).$$

Heun's method has order-2 local truncation error. Hence $e_n = \mathcal{O}(h^2)$.

Runge-Kutta Methods

Many one-step methods are part of a broad family called Runge-Kutta methods (Carl Runge 1895, Wilhelm Kutta 1901).

An s-stage explicit Runge-Kutta method is defined by first setting

$$\begin{aligned}
k_1 &= f(t_k, u_k), \\
k_2 &= f(t_k + c_2 h, u_k + h a_{2,1} k_1), \\
k_3 &= f(t_k + c_3 h, u_k + h a_{3,1} k_1 + h a_{3,2} k_2), \\
&\vdots \qquad \vdots \\
k_s &= f(t_k + c_s h, u_k + h a_{s,1} k_1 + \cdots + h a_{s,s-1} k_{s-1}).
\end{aligned}$$

Then our method is

$$u_{k+1} = u_k + h(b_1 k_1 + \cdots b_s k_s).$$

The values $a_{i,j}$, c_i and b_i are coefficients which determine the specific Runge-Kutta method.

Runge-Kutta Methods

Explicit Euler and Heun's method are both Runge-Kutta methods (check!).

The most famous Runge-Kutta method is the classical 4th order method RK4:

$$
\begin{aligned}
k_1 &= f(t_k, u_k), \\
k_2 &= f\left(t_k + \frac{h}{2}, u_k + \frac{h}{2}k_1\right), \\
k_3 &= f\left(t_k + \frac{h}{2}, u_k + \frac{h}{2}k_2\right), \\
k_4 &= f(t_k + h, u_k + hk_3),
\end{aligned}
$$

and

$$
u_{k+1} = u_k + h\left(\frac{1}{6}k_1 + \frac{2}{6}k_2 + \frac{2}{6}k_3 + \frac{1}{6}k_4\right).
$$

If $f(t, u)$ depends only on t, this corresponds to Simpson's rule for quadrature on $[t_k, t_{k+1}]$.

Runge-Kutta Methods

Runge-Kutta methods can also be implicit (requiring a rootfinding method to calculate the k_i values).

A general *s*-stage Runge-Kutta method is defined by first setting

$$\begin{aligned}
k_1 &= f(t_k + c_1 h, u_k + h a_{1,1} k_1 + \cdots + h a_{1,s} k_s),\\
k_2 &= f(t_k + c_2 h, u_k + h a_{2,1} k_1 + \cdots + h a_{2,s} k_s),\\
k_3 &= f(t_k + c_3 h, u_k + h a_{3,1} k_1 + \cdots + h a_{3,s} k_s),\\
&\vdots \qquad \vdots\\
k_s &= f(t_k + c_s h, u_k + h a_{s,1} k_1 + \cdots + h a_{s,s} k_s).
\end{aligned}$$

and then calculating

$$u_{k+1} = u_k + h(b_1 k_1 + \cdots b_s k_s).$$

Runge-Kutta Methods

For systems of ODEs, $\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y})$, Runge-Kutta methods are exactly the same (just with vectors everywhere):

$$\begin{aligned}
\boldsymbol{k}_1 &= f(t_k + c_1 h, \boldsymbol{u}_k + h a_{1,1}\boldsymbol{k}_1 + \cdots + h a_{1,s}\boldsymbol{k}_s),\\
\boldsymbol{k}_2 &= f(t_k + c_2 h, \boldsymbol{u}_k + h a_{2,1}\boldsymbol{k}_1 + \cdots + h a_{2,s}\boldsymbol{k}_s),\\
\boldsymbol{k}_3 &= f(t_k + c_3 h, \boldsymbol{u}_k + h a_{3,1}\boldsymbol{k}_1 + \cdots + h a_{3,s}\boldsymbol{k}_s),\\
&\vdots \qquad \vdots\\
\boldsymbol{k}_s &= f(t_k + c_s h, \boldsymbol{u}_k + h a_{s,1}\boldsymbol{k}_1 + \cdots + h a_{s,s}\boldsymbol{k}_s).
\end{aligned}$$

and then

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + h(b_1\boldsymbol{k}_1 + \cdots b_s\boldsymbol{k}_s).$$

For implicit methods, finding the $\boldsymbol{k}_i$ values requires a multi-dimensional rootfinding method.

The same simple extension to vectors also works for multistep methods (later).

Runge-Kutta Methods

For simplicity, Runge-Kutta methods are sometimes written as a Butcher tableau (John Butcher):

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,s} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

If the c_i are sorted and the matrix of $a_{i,j}$ is strictly lower triangular, the method is explicit.

For example, Heun's method corresponds to the Butcher tableau

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

Gauss Collocation

Gaussian Quadrature

We learned in the integration lectures that Gaussian quadrature gives the highest-order integration rules. Can we use these to build ODE methods?

Yes, and they all give Runge-Kutta methods! The simplest is the implicit midpoint rule

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array} \quad \text{or} \quad \begin{aligned} k_1 &= f(t_k + h/2, u_k + hk_1/2), \\ u_{k+1} &= u_k + hk_1. \end{aligned}$$

After rearranging, this is

$$u_{k+1} = u_k + hf\left(t_k + \frac{h}{2}, \frac{u_k + u_{k+1}}{2}\right).$$

In general however, Gauss quadrature-based rules are not widely used for ODEs because they are implicit and too slow to calculate at each timestep.

Order of RK Methods

We can figure out requirements on RK schemes to guarantee particular orders for the local truncation error. For example, consider the generic 2-stage explicit scheme:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c & a & 0 \\ \hline & b_1 & b_2 \end{array}$$

We will check the order by doing a Taylor expansion of

$$L(t,h) = \frac{u(t+h)-u(t)}{h} - [b_1 f(t,u(t)) + b_2 f(t+ch, u(t)+ahf(t,u(t))].$$

For this, we will need the multivariate chain rule:

$$\text{if } z = f(x(t),y(t)), \quad \text{then} \quad \frac{dz}{dt} = \frac{\partial z}{\partial x}\frac{dx}{dt} + \frac{\partial z}{\partial y}\frac{dy}{dt}$$

Order of RK Methods

The first term has Taylor expansion:

$$\begin{aligned}\frac{u(t+h)-u(t)}{h} &= u'(t) + \frac{h}{2}u''(t) + \mathcal{O}(h^2),\\ &= f(t,u(t)) + \frac{h}{2}\underbrace{\left[f_t(t,u(t)) + f_u(t,u(t))f(t,u(t))\right]}_{=df/dt = d(u')/dt} + \mathcal{O}(h^2).\end{aligned}$$

The second term is a bit trickier:

$$\begin{aligned}&b_1 f(t,u(t)) + b_2 f(t+ch, u(t)+ahf(t,u(t))\\ &\quad= b_1 f(t,u(t)) + b_2\left[f(t,u(t)) + chf_t(t,u(t)) + ahf(t,u(t))f_u(t,u(t)) + \mathcal{O}(h^2)\right],\\ &\quad= (b_1+b_2)f(t,u(t)) + h\left[b_2 c f_t(t,u(t)) + b_2 a f_u(t,u(t))f(t,u(t))\right] + \mathcal{O}(h^2).\end{aligned}$$

Order of RK Methods

Combining these, we get

$$L(t,h) = (1-(b_1+b_2))f(t,u(t)) + h\left[\left(\frac{1}{2}-b_2c\right)f_t(t,u(t)) + \left(\frac{1}{2}-b_2a\right)f_u(t,u(t))f(t,u(t))\right] + \mathcal{O}(h^2).$$

So, the method is second order (i.e. $L(t,h)=\mathcal{O}(h^2)$) if

$$b_1+b_2=1, \qquad b_2c=\frac{1}{2}, \quad \text{and} \quad b_2a=\frac{1}{2}.$$

Heun's method meets this conditions, for example ($b_1=b_2=\frac{1}{2}$, $a=c=1$).

Comparison of Explicit RK Methods

Let's compare 4 explicit RK methods:

- Explicit Euler:

$$u_{k+1} = u_k + hf(t_k, u_k).$$

- Explicit midpoint rule:

$$u_{k+1} = u_k + hf\left(t_k + \frac{h}{2}, u_k + \frac{h}{2}f(t_k, u_k)\right).$$

- Heun's method:

$$u_{k+1} = u_k + \frac{h}{2}f(t_k, u_k) + \frac{h}{2}f(t_k + h, u_k + hf(t_k, u_k)).$$

- RK4:

$$u_{k+1} = u_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Comparison of Explicit RK Methods

$$\frac{du}{dt} = -4t(1+t^2)u^2, \text{ with } u(0) = 1 \qquad \Longleftrightarrow \qquad u(t) = \frac{1}{(t^2+1)^2}$$

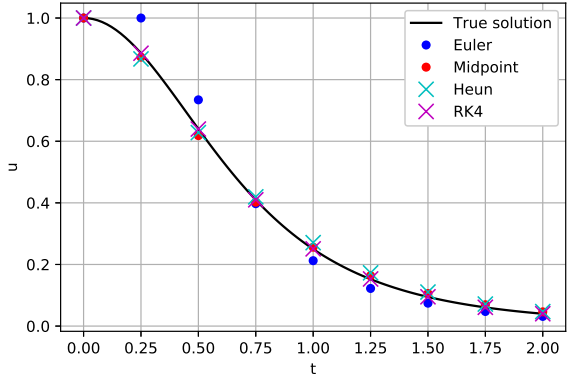

Figure 4. Explicit Method comparison with $h = 0.25$ on $[0, 2]$.

Comparison of Explicit RK Methods

$$\frac{du}{dt} = -4t(1+t^2)u^2, \text{ with } u(0) = 1 \qquad \Longleftrightarrow \qquad u(t) = \frac{1}{(t^2+1)^2}$$

Now look at maximum error at any time step in for decreasing values of h:

h	Euler	Midpoint	Heun	RK4
2.0e-01	9.043710e-02	1.248089e-02	1.322029e-02	2.763936e-04
2.0e-02	7.420119e-03	8.596333e-05	1.022094e-04	2.131151e-08
2.0e-03	7.245335e-04	8.309042e-07	9.956739e-07	2.061351e-12
2.0e-04	7.228165e-05	8.281259e-09	9.931226e-09	1.998401e-15
2.0e-05	7.226451e-06	8.278267e-11	9.928486e-11	1.010303e-14

- Euler is $\mathcal{O}(h)$, midpoint and Heun are $\mathcal{O}(h^2)$
- RK4 is $\mathcal{O}(h^4)$ (until machine precision approximately reached).

Stability

Stability of ODEs:

- A solution to an ODE is unstable if small perturbations to the initial conditions grow to infinity over time. For example, $u'(t) = u$ has solutions $u(t) = u_0 e^t$.
- A solution is stable if small perturbations remain bounded over time (do not grow to infinity). For example, $u'(t) = -u$ has solutions $u(t) = u_0 e^{-t}$.
- A solution is asymptotically stable if small perturbations go to zero over time.

Stability of our numerical methods: do our numerical errors grow over time or not?

- We obviously want them to stay small (ideally to go zero).
- If our method is convergent (consistent and stable), then the errors go to zero as $h \to 0$.
  - All done?
- This is a limiting result. What happens when we choose a fixed h?
  - We don't know whether the limiting behaviour applies or not!

Perturbation Analysis

Suppose we have the ODE $\frac{du}{dt} = f(t, u)$, and an original solution $u_0(t)$.

We then perturb the initial conditions by ϵ and get a new ODE solution $u_\epsilon(t)$. Assume that (up to a Taylor series-type error)

$$u_\epsilon(t) = u_0(t) + \epsilon v(t).$$

Therefore

$$\frac{du_\epsilon}{dt} = \frac{du_0}{dt} + \epsilon \frac{dv}{dt}.$$

Also, by Taylor series,

$$f(t, u_\epsilon(t)) = f(t, u_0(t)) + f_u(t, u_0(t))\epsilon v(t) + \mathcal{O}(\epsilon^2).$$

Comparing the terms of size ϵ, we see that the perturbation $v(t)$ approximately satisfies

$$\frac{dv}{dt} \approx f_u(t, u_0(t))v(t).$$

The perturbation is asymptotically stable if $\lim_{t\to\infty} v(t) = 0$.

Dahlquist Test Problem

We derived the ODE: $\frac{dv}{dt} \approx [f_u(t, u_0(t))]v$.

Since perturbations satisfy this ODE, when we study numerical methods we consider the Dahlquist test problem (Germund Dahlquist, mid-late 20th century)

$$\boxed{\frac{du}{dt} = \lambda u, \qquad u(0) = 1}$$

where $\lambda \in \mathbb{C}$ is a constant. The solution is

$$u(t) = e^{\lambda t},$$

and so (for thinking about asymptotic stability)

$$\lim_{t\to\infty} u(t) = 0 \qquad \text{if and only if} \qquad \text{Re}(\lambda) < 0,$$

where $\text{Re}(\cdot)$ is the real part of a complex number.

Dahlquist Test Problem

$$\boxed{\frac{du}{dt} = \lambda u, \qquad u(0) = 1}$$

Now let's run explicit Euler on this ODE:

$$u_0 = 1, \qquad u_{k+1} = u_k + h\lambda u_k = (1 + h\lambda)u_k.$$

We write $u_{k+1} = \rho(h\lambda)u_k$, where here $\rho(z) = 1 + z$ is the amplification factor.

Note that $1 + h\lambda$ is a first-order Taylor series approximation for $e^{h\lambda}$.

When does our solution $u_k \to 0$? We get

$$|u_k| = |\rho(h\lambda)|\,|u_{k-1}| = |\rho(h\lambda)|^2|u_{k-2}| = \cdots = |\rho(h\lambda)|^k \underbrace{|u_0|}_{=1}.$$

Hence $u_k \to 0$ whenever $|\rho(h\lambda)| < 1$. That is, when $|1 + h\lambda| < 1$.

Dahlquist Test Problem

The explicit Euler solution converges to 0 when $h\lambda$ satisfy $|1 + h\lambda| < 1$.

This defines a region in the complex plane: a circle with centre -1, radius 1.

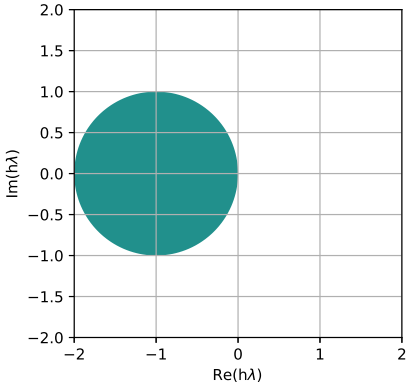

Figure 5. Region of A-stability for explicit Euler.

Dahlquist Test Problem

Suppose $\lambda < 0$ is a real number with $u(t) \to 0$. Then $|1 + h\lambda| < 1$ means

$$-1 < 1 + h\lambda < 1, \qquad \text{or} \qquad 0 < h < \frac{2}{|\lambda|}.$$

So if we want $u_k \to 0$, the value of λ puts restrictions on the size of h.

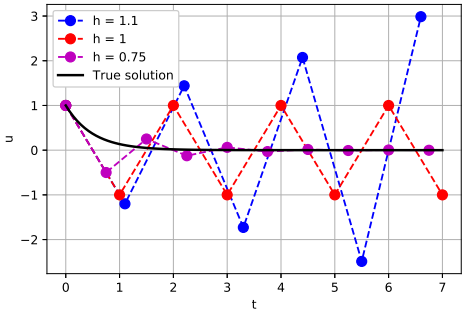

Figure 6. Explicit Euler with $\lambda = -2$ for different values of h.

A-Stability

Definition

The region of A-stability of a numerical method with amplification factor $\rho(z)$ is the set

$$\{z \in \mathbb{C} : |\rho(z)| < 1\}.$$

A method is called unconditionally stable (sometimes just "A-stable") if all points with $\text{Re}(z) < 0$ are in the region of A-stability (i.e. the stability region contains the left-hand half of the Argand diagram).

If a method is unconditionally stable, then $u_k \to 0$ whenever $\text{Re}(\lambda) < 0$, regardless of the value of h.

The region of A-stability for explicit Euler was shown on the previous slide. It is not unconditionally stable.

Implicit Euler

The most basic implicit scheme is the implicit Euler method:

$$u_{k+1} = u_k + hf(t_{k+1}, u_{k+1}).$$

It corresponds to quadrature with right Riemann sums. To implement it, we need to use a rootfinding method to find u_{k+1} at each iteration.

Applied to the Dahlquist problem, implicit Euler is

$$u_{k+1} = u_k + h\lambda u_{k+1}, \qquad \text{and so} \qquad u_{k+1} = \frac{1}{1-h\lambda} u_k = \rho(h\lambda) u_k.$$

The amplification factor for implicit Euler is $\rho(z) = 1/(1-z)$.

So, $|\rho(h\lambda)| < 1$ whenever $|1 - h\lambda| > 1$. This is the exterior of a circle with centre 1, radius 1.

Note: the amplification factor for explicit methods is a polynomial. For implicit methods it is a rational function (ratio of polynomials).

Implicit Euler

The region of A-stability for implicit Euler, where $\rho(h\lambda) = 1/(1-h\lambda)$, is

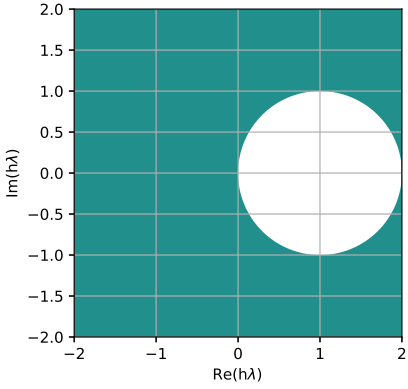

Figure 7. Region of A-stability for implicit Euler.

This contains the left-hand half of the plane, so implicit Euler is unconditionally stable.

Implicit Euler

Since implicit Euler is unconditionally stable, for $u_k \to 0$ (with $\text{Re}(\lambda) < 0$) there are no restrictions on h:

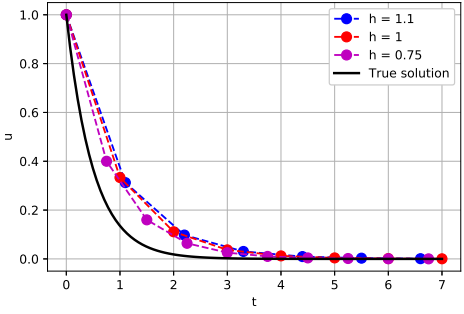

Figure 8. Implicit Euler with $\lambda = -2$ for different values of h.

Warning! Just because there are no restrictions on h for $u_k \to 0$, doesn't mean the accuracy (size of errors) is the same: small h are still more accurate.

A-Stability of RK4

What is the A-stability of RK4? Applied to the Dahlquist problem, we get

$$
\begin{aligned}
k_1 &= f(t_k, u_k) = \lambda u_k, \\
k_2 &= f\left(t_k + \frac{h}{2}, u_k + \frac{h}{2}k_1\right) = \lambda(u_k + \frac{h}{2}k_1) = \lambda u_k + \frac{h}{2}\lambda^2 u_k, \\
k_3 &= f\left(t_k + \frac{h}{2}, u_k + \frac{h}{2}k_2\right) = \lambda(u_k + \frac{h}{2}k_2) = \lambda u_k + \frac{h}{2}\lambda^2 u_k + \frac{h^2}{4}\lambda^3 u_k, \\
k_4 &= f(t_k + h, u_k + hk_3) = \lambda(u_k + hk_3) = \lambda u_k + h\lambda^2 u_k + \frac{h^2}{2}\lambda^3 u_k + \frac{h^3}{4}\lambda^4 u_k, \\
u_{k+1} &= u_k + h\left(\frac{1}{6}k_1 + \frac{2}{6}k_2 + \frac{2}{6}k_3 + \frac{1}{6}k_4\right) = \left[1 + h\lambda + \frac{h^2\lambda^2}{2} + \frac{h^3\lambda^3}{6} + \frac{h^4\lambda^4}{24}\right] u_k.
\end{aligned}
$$

Hence the amplification factor for RK4 is the polynomial

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

This is a 4th-order Taylor series approximation to e^z.

A-Stability of RK4

The region of A-stability for RK4, where $\rho(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24}$, is

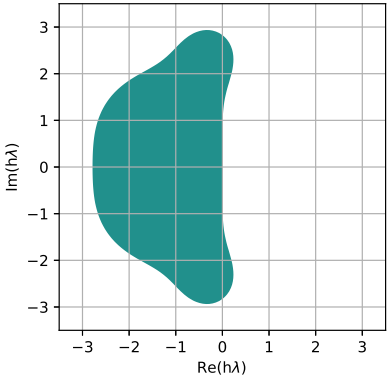

Figure 9. Region of A-stability for RK4.

Hence RK4 is not unconditionally stable (but it is still 4th-order accurate).

Stiff ODEs

An important category of ODEs (from a numerical point-of-view) are so-called stiff problems. There is no standard definition, but we informally say an ODE is stiff when:

- The solution is changing very quickly (compared to the time interval being considered).
- For a system of ODEs, when one component of the solution $\boldsymbol{u}(t)$ varies much more than another component.

In general, for stiff problems we need to set h very small to ensure stability.

Generally, implicit methods are much better for stiff problems, since they generally allow larger values of h to be used (fewer calculations needed).

Stiff ODE: Robertson System

An example of a stiff ODE system is the Robertson system (1966) which model autocatalytic chemical reactions:

$$A \to B,$$
$$2B \to B + C,$$
$$B + C \to A + C.$$

Writing the concentrations of A, B and C as $u_1(t)$, $u_2(t)$ and $u_3(t)$, we get the ODE system

$$\frac{du_1}{dt} = -0.04u_1 + 10^4 u_2 u_3,$$
$$\frac{du_2}{dt} = 0.04u_1 - 3 \times 10^7 u_2^2 - 10^4 u_2 u_3,$$
$$\frac{du_3}{dt} = 3 \times 10^7 u_2^2.$$

We will use the initial conditions $\boldsymbol{u}_0 = (1, 0, 0)$ and solve for $t \in [0, 0.1]$.

Stiff ODE: Robertson System

Using explicit Euler with time step $h = 10^{-4}$ we get the solution

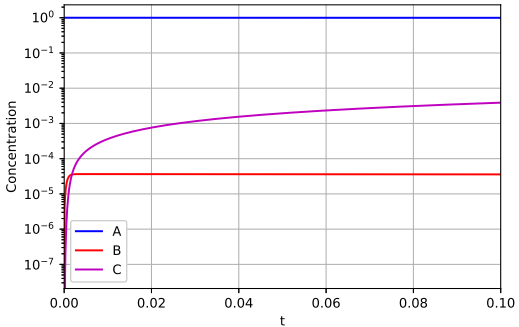

Figure 10. Explicit Euler for Robertson system ($h = 10^{-4}$).

Stiff ODE: Robertson System

Now using explicit Euler with time step $h = 10^{-3}$ we get the solution

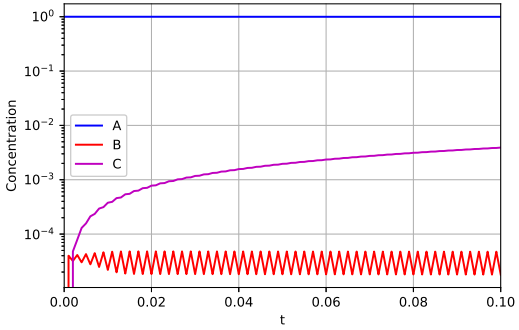

Figure 11. Explicit Euler for Robertson system ($h = 10^{-3}$).

Stiff ODE: Robertson System

RK4 is an improvement, but we still get stability issues at $h \approx 1.6 \times 10^{-3}$:

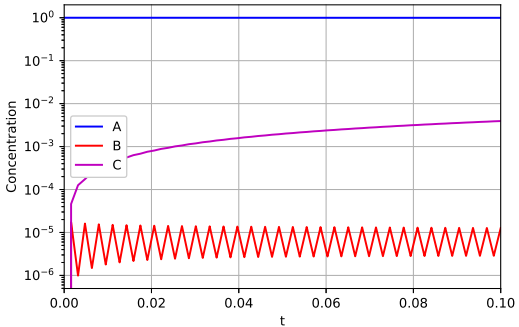

Figure 12. RK4 for Robertson system ($h \approx 1.6 \times 10^{-3}$).

Stiff ODE: Robertson System

However implicit Euler (1st order method) works fine even for $h = 0.01$:

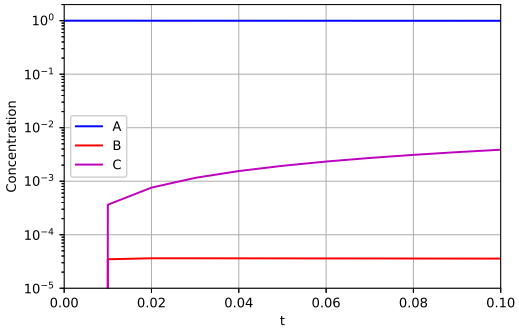

Figure 13. Implicit Euler for Robertson system ($h = 0.01$).

Expensive Evaluations

When we looked at quadrature, we looked at two categories of methods of the form $\int_a^b f(x)dx \approx \sum_{i=0}^n c_i f(x_i)$

- Gauss quadrature: pick x_i and c_i to give highest-order method
- Newton-Cotes: given fixed x_i (e.g. equally spaced), pick best c_i.

Newton-Cotes are useful when evaluations of $f(x)$ are hard to get (run an experiment, slow computation, etc.).

Runge-Kutta methods are similar to Gaussian quadrature methods: they require evaluations of $f(t,u)$ at many different combinations of t and u. For example, RK4 uses 4 evaluations of $f(t,u)$ to get from u_k to u_{k+1}.

Question

What if evaluations of $f(t,u)$ are expensive? Can we build methods which only use previous evaluations of $f(t_k,u_k)$ to calculate u_{k+1}?

Example Multistep Method

Let's look at a simple example. Suppose we are looking for a method of the form

$$u_{k+1} = \alpha_1 u_k + h\left(\beta_1 f(t_k, u_k) + \beta_2 f(t_{k-1}, u_{k-1})\right)$$

We find the unknowns using the method of undetermined coefficients.

We ask our method to be exact whenever the true solution $u(t)$ is a polynomial of degree ≤ 2 (since there are 3 unknowns). Let's make the rule exact for $u(t) = 1, t, t^2$ (i.e. $f(t, u) = 0, 1, 2t$):

$$\begin{aligned}
1 &= \alpha_1 \cdot 1 + h\left(\beta_1 \cdot 0 + \beta_2 \cdot 0\right), \\
t_{k+1} &= \alpha_1 t_k + h\left(\beta_1 \cdot 1 + \beta_2 \cdot 1\right), \\
t_{k+1}^2 &= \alpha_1 t_k^2 + h\left(\beta_1 \cdot 2t_k + \beta_2 \cdot 2t_{k-1}\right).
\end{aligned}$$

We want these equations to work for any choice of t_k and h, so let's just pick $h = 1$ and $t_{k-1} = 0,\ t_k = 1,\ t_{k+1} = 2$.

Example Multistep Method

Substituting $h = 1$, $t_{k-1} = 0$, $t_k = 1$, $t_{k+1} = 2$ we get

$$\begin{aligned} 1 &= \alpha_1 + 0\beta_1 + 0\beta_2, \\ 2 &= \alpha_1 + \beta_1 + \beta_2, \\ 4 &= \alpha_1 + 2\beta_1 + 0\beta_2, \end{aligned}$$

with solution $\alpha_1 = 1$, $\beta_1 = 3/2$, $\beta_2 = -1/2$. Hence our method is

$$u_{k+1} = u_k + \frac{3h}{2} f(t_k, u_k) - \frac{h}{2} f(t_{k-1}, u_{k-1}),$$

which works for equally spaced grids $t_k = t_0 + kh$.

This is a two-term recursion (u_{k+1} needs u_k and u_{k-1}), so we need two initial values u_0 and u_1.

- We can get u_0 from the ODE initial conditions.
- The easiest way to get u_1 is to use a one-step method (e.g. one iteration of explicit Euler).

General Linear Multistep Methods

A linear multistep method is a method of the form:

$$\alpha_m u_{k+m} + \alpha_{m-1} u_{k+m-1} + \cdots + \alpha_0 u_k = h\left(\beta_m f_{k+m} + \beta_{m-1} f_{k+m-1} + \cdots + \beta_0 f_k\right),$$

$$\sum_{j=0}^{m} \alpha_j u_{k+j} = h\left(\sum_{j=0}^{m} \beta_j f_{k+j}\right),$$

where $f_{k+j} = f(t_{k+j}, u_{k+j})$. Here, we find u_{k+m} given $u_k, \ldots, u_{k+m-1}$, where m is the number of steps.

We assume that

- $\alpha_m \neq 0$, so we can solve for u_{k+m}, and
- $\alpha_0^2 + \beta_0^2 \neq 0$, so we have the correct value of m.

Note that a linear m-step method is explicit if $\beta_m = 0$.

General Linear Multistep Methods

$$\boxed{\sum_{j=0}^{m} \alpha_j u_{k+j} = h\left(\sum_{j=0}^{m} \beta_j f_{k+j}\right)}$$

For example, explicit Euler is a multistep method with $m = 1$, $(\alpha_0, \alpha_1) = (-1, 1)$ and $(\beta_0, \beta_1) = (1, 0)$:

$$u_{k+1} - u_k = hf(t_k, u_k).$$

The multistep method we derived earlier (after re-indexing)

$$u_{k+2} = u_{k+1} + \frac{3h}{2} f(t_{k+1}, u_{k+1}) - \frac{h}{2} f(t_k, u_k),$$

corresponds to $m = 2$, $(\alpha_0, \alpha_1, \alpha_2) = (0, -1, 1)$ and $(\beta_0, \beta_1, \beta_2) = (-1/2, 3/2, 0)$.

Important Multistep Methods

Adams-Bashforth methods (John Couch Adams, 1855, Francis Bashforth, 1883):

- Choose $\alpha_m = 1$, $\alpha_{m-1} = -1$ and $\alpha_{m-2}, \ldots, \alpha_0 = 0$.
- Set $\beta_m = 0$ (explicit) and $\beta_{m-1}, \ldots, \beta_0$ so that the method has the highest possible order.

The methods for $m = 1$ and $m = 2$ are familiar:

$$u_{k+1} - u_k = hf_k,$$
$$u_{k+2} - u_{k+1} = \frac{3h}{2} f_{k+1} - \frac{h}{2} f_k.$$

If $m = 4$, for example, we get the 4th-order method

$$u_{k+4} - u_{k+3} = \frac{h}{24} \left(55f_{k+3} - 59f_{k+2} + 37f_{k+1} - 9f_k\right).$$

Important Multistep Methods

Adams-Moulton methods (Forest Ray Moulton, c. 1915):

- Choose $\alpha_m = 1$, $\alpha_{m-1} = -1$ and $\alpha_{m-2}, \ldots, \alpha_0 = 0$.
- Set $\beta_m, \ldots, \beta_0$ so that the method has the highest possible order.

The methods are implicit ($\beta_m \neq 0$).

For example $m = 1$ is the implicit trapezoidal rule:

$$u_{k+1} - u_k = \frac{h}{2}\left(f_{k+1} + f_k\right).$$

If $m = 4$, for example, we get the 5th-order method

$$u_{k+4} - u_{k+3} = \frac{h}{720}\left(251 f_{k+4} + 646 f_{k+3} - 264 f_{k+2} + 106 f_{k+1} - 19 f_k\right).$$

Important Multistep Methods

Backwards differentiation formulae:

- Choose $\alpha_m = 1$ and $\beta_{m-1}, \ldots, \beta_0 = 0$.
- Set $\alpha_{m-1}, \ldots, \alpha_0$ and β_m so that the method has the highest possible order.

The methods are implicit ($\beta_m \neq 0$).

For example, $m = 1$ gives backwards Euler

$$u_{k+1} - u_k = h f_{k+1}.$$

If $m = 4$, for example, we get the 4th-order method

$$u_{k+4} - \frac{48}{25} u_{k+3} + \frac{36}{25} u_{k+2} - \frac{16}{25} u_{k+1} + \frac{3}{25} u_k = \frac{12h}{25} f_{k+4}.$$

Multistep Methods: Consistency

The consistency of a multistep method is analysed similar to one-step methods. We look at the error between $u(t_{k+m})$ and u_{k+m} calculated using the correct values for $u(t_{k+m-1}), \dots, u(t_k)$.

That is, if u_{k+m} is calculated using the correct values for all other u_{k+j} and f_{k+j} we define

$$L(t_{k+m-1}, h) := \frac{u(t_{k+m}) - u_{k+m}}{h}$$

Theorem

A linear multistep method has consistency order p ($L = \mathcal{O}(h^p)$ for all k) if

$$\sum_{j=0}^{m} \alpha_j = 0, \quad \sum_{j=0}^{m} \beta_j \neq 0, \quad \text{and} \quad \sum_{j=0}^{m} \alpha_j \cdot j^q = q \sum_{j=0}^{m} \beta_j \cdot j^{q-1}, \qquad \forall q = 1, \dots, p.$$

Note: for $q = 1$, we use the convention $j^{q-1} = 1$ for all j, including $j = 0$.

A method is consistent if it has consistency order $p \geq 1$.

Multistep Methods: Zero Stability

The important notion of stability for multistep methods is zero stability: that is, if we start our multistep method with slightly perturbed initial data $u_0, \ldots, u_m$ (of size ϵ), the calculated solution does not change by more than $\mathcal{O}(\epsilon)$.

Theorem

A linear multistep method is zero stable if it is consistent and all the (complex) roots of the polynomial

$$\rho(z) := \sum_{j=0}^{m} \alpha_j z^j,$$

satisfy $|z| \leq 1$, and all roots with $|z| = 1$ are simple (i.e. not double roots, etc.).

Multistep Methods: Convergence

Just like for one-step methods, consistency and stability give us convergence. In this case, a multistep method is convergent if $\lim_{h\to 0} u_N = u(T)$ (where $N = (T - t_0)/h$), for any choice of initial data with $u_0, \ldots, u_m \to u(0)$ as $h \to 0$.

Theorem

A linear multistep method is convergent if and only if it is consistent and zero stable.

If the method has consistency order p and the initial data satisfies $|u(t_s) - u_s| = \mathcal{O}(h^p)$ for $s = 0, \ldots, m$, then $|u_k - u(t_k)| = \mathcal{O}(h^p)$ for all $k = 0, \ldots, N$.

Theorem

If a linear m-step method is zero stable, then its consistency order p satisfies:

- $p \leq m + 2$ *if m is even;*
- $p \leq m + 1$ *if m is odd;*
- $p \leq m$ *if $\beta_m/\alpha_m \leq 0$ (e.g. any explicit method).*

Adaptive Methods

Often we want to control the accuracy of our solution: i.e. supply some desired error tolerance to our algorithm (and finish when we have that level of accuracy).

It turns out we can do this for ODE methods. What we need to do is to vary the time step at each iteration:

$$u_{k+1} = u_k + h_k \phi(t_k, u_k),$$

for non-constant h_k.

- We can't control the true error $|u_k - u(t_k)|$ since we don't know $u(t_k)$.
- We can approximate the error (e.g. with a more accurate method), but if we need to change h_k then we need to recompute our high-accuracy method from the start.

Instead we try to control the local truncation error $L(t_k, h_k)$ by comparing our method to a higher-order method (as a proxy for the true solution).

Adaptive Methods

Suppose we have our 'base' scheme of interest (e.g. explicit Euler):

$$u_{k+1} = u_k + h\phi(t_k, u_k),$$

with order-p local truncation error

$$L(t_k, h_k) = \frac{u(t_{k+1}) - u(t_k)}{h_k} - \phi(t_k, u(t_k)) = \mathcal{O}(h_k^p).$$

Our higher-order method (e.g. Heun) is

$$\widetilde{u}_{k+1} = \widetilde{u}_k + h\widetilde{\phi}(t_k, \widetilde{u}_k).$$

It must be at least order $p+1$ (i.e. $\widetilde{L}(t_k, h_k) = \mathcal{O}(h_k^{p+1})$).

Adaptive Methods

Suppose we have been accurate so far, so $u_k \approx \widetilde{u}_k \approx u(t_k)$. Then the 'base' local truncation error can be approximated by:

$$L(t_k, h_k) = \frac{u(t_{k+1}) - u(t_k)}{h_k} - \phi(t_k, u(t_k)) \approx \frac{\widetilde{u}_{k+1} - u_k}{h_k} - \phi(t_k, u_k) = \frac{\widetilde{u}_{k+1} - u_{k+1}}{h_k}.$$

We can't use $u(t_{k+1}) \approx u_{k+1}$ because we would get zero, hence we need $\widetilde{u}_{k+1}$.

If $|L(t_k, h_k)|$ is too large, we instead try step qh_k for $q < 1$. Since the base method is order p:

$$L(t_k, qh_k) \approx Cq^p h_k^p = q^p L(t_k, h_k) \approx q^p \frac{\widetilde{u}_{k+1} - u_{k+1}}{h_k}.$$

So, if we want $|L(t_k, qh_k)| \leq \epsilon$, we reduce h_k by a factor

$$q \leq \left(\frac{\epsilon h_k}{|\widetilde{u}_{k+1} - u_{k+1}|} \right)^{1/p}.$$

We repeat this process and reduce h_k until we get $|L| \leq \epsilon$, then take the step and proceed to the next iteration.

Adaptive Methods

The most common way to build the base and higher-order methods are embedded Runge-Kutta methods. These are two RK methods with the same c_i and $a_{i,j}$, but different b_i.

This reduces the number of times we need to evaluate $f(t, u)$.

Example: explicit Euler and Heun have Butcher tableaux which can be written as

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

The default ODE solver in SciPy uses embedded RK methods (4th + 5th order, both 6-stage explicit methods), often abbreviated to RK45.

Other methods are available too. See the documentation for `scipy.integrate.solve_ivp`.

Structure-Preserving Methods

Some ODEs have solutions where particular relationships are preserved (e.g. conservation of energy). Often in these cases we want our numerical solution to preserve the same quantities.

For example, the ODE system

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \begin{bmatrix} -\sin t \\ \cos t \end{bmatrix}, \text{ with } \begin{bmatrix} x(0) \\ y(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

has the solution $x(t) = \cos t$ and $y(t) = \sin t$. Hence

$$x(t)^2 + y(t)^2 = 1, \qquad \forall t.$$

Structure-Preserving Methods

Running explicit Euler and implicit midpoint methods (with the same timestep), we get:

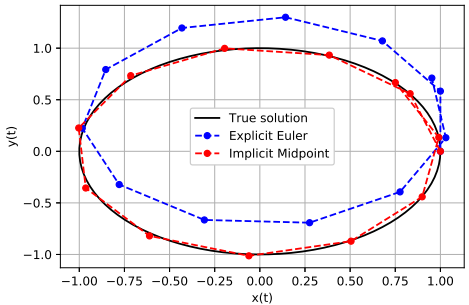

Figure 14. Calculated solutions for $x(t)$ and $y(t)$.

Here, explicit Euler does not preserve $x_k^2 + y_k^2 = 1$, but the implicit midpoint rule does. Methods which preserve quantities of interest are sometimes called structure-preserving methods or geometric integrators.

Boundary Value Problems

What about boundary value problems? These are much more complicated to study mathematically. For example, non-uniqueness of solutions:

$$u''(t) + u(t) = 0, \qquad u(0) = 1, \qquad u(1) = -1,$$

has multiple solutions

$$u(t) = \cos(\pi t) + \alpha \sin(\pi t),$$

for any $\alpha \in \mathbb{R}$.

However, if the second condition was $u(1) = 0$, the BVP would have no solution!

Other types of conditions also arise from time-to-time (e.g. $\int_0^1 u(t)dt = 1$) which have similar complexities.

Solving BVPs

The easiest way to solve BVPs is the shooting method.

Here, we only impose initial conditions, then find some which make the other boundary condition true. For example, consider the Bratu equation

$$u''(t) + 3e^{u(t)} = 0, \qquad u(0) = u(1) = 0.$$

Instead we solve the family of IVPs

$$u''(t) + 3e^{u(t)} = 0, \qquad u(0) = 0, \qquad u'(0) = v_0,$$

and calculate the value $u(1)$ of the solution (which will depend on v_0).

Then, use a rootfinding method to find value(s) of v_0 which give $u(1) = 0$.

Shooting Method: Example

Calculating $u(1)$ for the IVP

$$u''(t) + 3e^{u(t)} = 0, \qquad u(0) = 0, \qquad u'(0) = v_0,$$

as a function of v_0 we get:

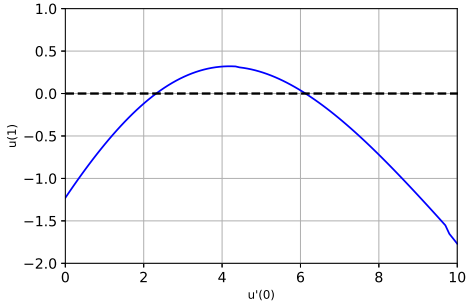

Figure 15. Plot of $u(1)$ vs. v_0.

Shooting Method: Example

Using a rootfinding algorithm, we find the roots are at $v_0 \approx 2.318, 6.129$. The two solutions to the IVP are:

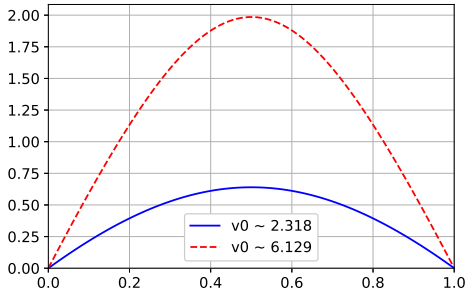

Figure 16. Two solutions to the Bratu equation.

Partial Differential Equations

An equation involving the derivatives of the function of multiple variables is called a partial differential equation (PDE). Some common examples include:

$$
\begin{aligned}
u_{xx} + u_{yy} &= f(x, y), && \text{(Poisson equation)} \\
u_t &= \kappa u_{xx} + f(x), && \text{(heat/diffusion equation)} \\
u_{tt} &= c^2 u_{xx}, && \text{(wave equation)} \\
u_t + u \cdot u_x &= \nu u_{xx}. && \text{(Burger's equation)}
\end{aligned}
$$

PDEs arise frequently in many areas of engineering, physics, chemistry, biology, finance, etc.

The theory of PDEs (when does a solution exist) and techniques for finding solutions are much harder than for ODEs. This is covered in MATH2306 and MATH4202.

Method of Lines

Some solution techniques for PDEs involve reducing the PDE into ODEs which we know how to solve. Similarly, we can solve some PDEs using numerical ODE methods (the method of lines).

For example, consider the heat equation with 1 spatial dimension (e.g. a bar being heated):

$$\begin{aligned} \frac{\partial u}{\partial t} &= \kappa \frac{\partial^2 u}{\partial x^2} + f(x,t), \qquad \forall x \in (0,L), t > 0, \\ u(x,0) &= u_{\text{init}}(x), \qquad \forall x \in [0,L], \\ u(0,t) = u(L,t) &= 0, \qquad \forall t > 0. \end{aligned}$$

For PDEs, we need to specify more extra conditions (here an initial condition for the temperature in the bar $u_{\text{init}}(x)$, and the temperature on the boundaries).

The term $f(x,t)$ represents any extra heat energy being added to the bar (e.g. a blowtorch), and κ is the thermal diffusivity of the bar (a physical property of the material).

Method of Lines

For the method of lines, we only look at the temperature at specific points along the bar:

$$\boldsymbol{u}(t) = \begin{bmatrix} u_0(t) \\ \vdots \\ u_N(t) \end{bmatrix},$$

where $u_j(t) = u(x_j, t)$ for points some $x_0 = 0$, ..., $x_N = L$ (e.g. equal grid size $h = L/n$).

The boundary conditions gives $u_0(t) = u_N(t) = 0$ and the initial conditions give $\boldsymbol{u}(0) = (u_{\text{init}}(x_0), \ldots, u_{\text{init}}(x_N))$.

We then approximate $\partial^2 u/\partial x^2$ using finite differences:

$$\frac{\partial^2 u}{\partial x^2}(x_j, t) \approx \frac{u(x_{j-1}, t) - 2u(x_j, t) + u(x_{j+1}, t)}{h^2}.$$

Method of Lines

All together, we get the ODE system

$$\begin{bmatrix} u_0'(t) \\ u_1'(t) \\ \vdots \\ u_{N-1}'(t) \\ u_N(t) \end{bmatrix} = \begin{bmatrix} 0 \\ \kappa(u_0(t) - 2u_1(t) + u_2(t))/h^2 + f(x_1, t) \\ \vdots \\ \kappa(u_{N-2}(t) - 2u_{N-1}(t) + u_N(t))/h^2 + f(x_{N-1}, t) \\ 0 \end{bmatrix},$$

with initial conditions $u_j(0) = u_{\text{init}}(x_j)$.

We can solve this using our numerical ODE techniques (but for this to be stable the timestep needs to be small enough; in this case we would usually need $\Delta t < h^2/(2\kappa)$).

This also works in higher dimensions: if we have a 2D being heated, we just discretise the block into a grid of points and look at $u_{i,j}(t) = u(x_i, y_j, t)$, for example.

Numerical Methods for PDEs

In general, solving PDEs numerically is not easy.

The simplest method is finite differencing (just use our standard techniques to approximate all the derivatives, then solve for u at grid points), but even that is complicated: e.g. grid spacing in different dimensions needs to be adjusted based on the PDE properties.

Other common techniques include spectral methods, finite element methods, and finite volume methods (but there are others too).

Main advice: talk to an expert! Specific information about the PDE can have a large impact on what techniques should be used (e.g. $u_{xx} + u_{yy} = 0$ is completely different to $u_{xx} - u_{yy} = 0$).

Developing, analysing and efficiently implementing PDE solvers is a very area of active ongoing research (at MSI and elsewhere).