Math 361S Lecture Notes
Numerical solution of ODEs: Part I

Jeffrey Wong

April 2, 2019

Topics covered

- Overview
  - Brief review of ODEs
  - Perspective for numerical methods
- Euler's method
  - Local and global truncation error; consistency
  - Convergence; proof of convergence
- Stability
  - Stiff differential equations
  - Regions of absolute stability
  - Some implicit methods (trapezoidal method, backward euler)
- Runge-Kutta methods
  - (Briefly) Taylor series method
  - Constructing higher-order one step methods
  - Runge-Kutta methods: the general approach
  - Example: RK4 (fixed step size)
- Implicit one step methods
  - General notes
  - Implementation

1 Overview of goals

In this section of the course we will consider the solution of ordinary differential equations. The tools we have developed so far - interpolation, differentiation and integtration - will serve as a foundation for constructing the algorithms. We will also make use of some linear algebra and non-linear equations (e.g. Newton's method) to solve some sub-problems.

Our main goal here twofold. First, we seek to understand the fundamentals of numerical methods for ODEs: what are the numerical issues that arise, how to recognize and correct them, and the right perspective. If you are handed a nasty ODE, how do you approach writing a method to solve it? How can you argue that the answer you obtain is correct?

Second, as ODEs are a synthesis of existing techniques, we will see how pieces can be combined and how the fundamental principles (error in interpolation, numerical stability of integration and so on) show up in practice. We'll also see a few more concepts:

- How stability manifests in numerical solution to ODEs (and how it connects to the mathematical notion of stability from theory)
- Managing trade-offs between accuracy and stability (the consequences of sacrificing one for the other are more severe for ODEs than the other problems we have seen)
- Designing adaptive algorithms that 'just work' like `ode45` in Matlab (we saw a bit of this with adaptive integration, but will explore it in detail here)

2 The problem to solve

2.1 Note on background

See sections 7.1-7.3 of Moler's book or any standard text on ODEs for a review of ODEs. The bare minimum will be presented here. Essentially no ODE theory is required to solve ODEs numerically, but the theory does provide important intuition, so it will greatly enhance your understanding of the numerics.

2.2 Initial value problems

Here, we will consider solving an **initial value problem**

$$y'(t) = f(t, y), \quad t \in [a, b] \tag{2.1a}$$

$$y(t_0) = y_0. \tag{2.1b}$$

The equation (2.1a) is the ODE (**ordinary differential equation**) for $y(t)$ and (2.1b) is the **initial condition**. We seek a function $y(t)$ that satisfies (2.1a) for all t in the given interval and whose value at t_0 is y_0.

The ODE is **first-order** (the highest derivative is first-order) and **scalar** ($y(t)$ is a real-valued function).

Throughout, we shall assume that

- $f(t, y)$ is a continuous function in t and y
- f has partial derivatives of all orders required for any derivation (mostly for Taylor expansions)

It is a fundamental theorem that these conditions ensure a unique solution exists, at least in some interval around (t_0, y_0). It is **not enough** to ensure that a solution exists for all t; the solution may diverge.

For our purposes, we will attempt to construct numerical solutions where the actual solution exists, so the theory is just there to ensure that the problem to solve is well-defined.

2.3 Slope fields and sensitivity

Here we review a useful geometric interpretation of the ODE

$$y' = f(t, y).$$

A solution $(t, y(t))$ forms a curve in the (t, y) plane. The ODE tells us the direction of the curve at any given point, since

$$y(t) \text{ is parallel to } (1, y') = (1, f).$$

In this sense, solutions to the ODE follow the 'slope field', which is the vector field

$$(1, f(t, y))$$

in the (t, y) plane. To find a solution to the IVP starting at (t_0, y_0), we may follow the slope field to construct the curve; this is the basis of the simplest numerical method that is detailed in the next section.

The slope field gives geometric intuition for some important concepts for numerical methods. For instance, we may ask: if $y(t)$ gets perturbed by some amount Δy at time t_0, how far apart are the original and perturbed curves after some time?

Put another way: how sensitive is the ODE to changes in the initial condition?

The picture is shown in Figure 3. Suppose we have two solutions $x(t)$ and $y(t)$ to the same ODE,

$$\begin{aligned} y' &= f(t, y), \quad y(t_0) = y_0, \\ x' &= f(t, x), \quad x(t_0) = x_0. \end{aligned}$$

Informally, the difference $z = y - x$ satisfies

$$z' = f(t, y) - f(t, x) = \frac{\partial f}{\partial y}(t, \xi) z$$

for some ξ, by the mean value theorem. Thus the size of $\partial f/\partial y$ determines how fast the difference can change with time. Let

$$L = \max_{t\in[a,b],y\in\mathbb{R}} \left|\frac{\partial f}{\partial y}(t,y)\right|.$$

Then we have, informally, that

$$|z|' \le L|z| \implies |z| \le |z_0|e^{Lt}$$

ignoring that the absolute value cannot be manipulated this way with a derivative. Thus L (the max. of the variation of f with respect to y) is the exponential rate, at worst, at which the two solutions can move apart.

However, the bound is sometimes pessimistic. Taking absolute values discards information about the sign, so if $z \sim -Lz$ then the bound is the same, even though z then decays exponentially. This is shown in the figure.

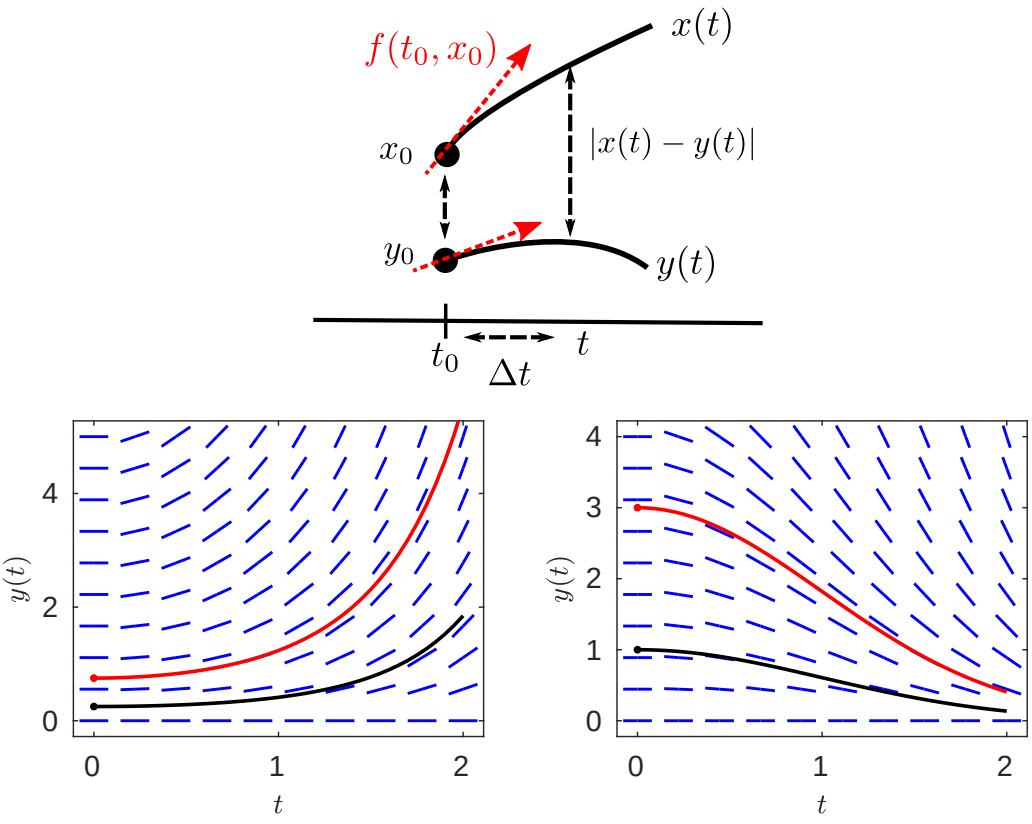


Figure 1: Sketch of the difference in two solutions that start at nearby points (t_0, x_0) and (t_0, y_0) and numerical examples for $y' = ty$ and $y' = -ty$.

3 Numerical methods: the basics

Summary of notation in this section:

- $t_0, \cdots, t_N$: the points where the approximate solution is defined
- $y_j = y(t_j)$: the exact solution to the IVP
- $\tilde{y}_j$ the approximation at t_n
- τ_j: truncation error in obtaining $\tilde{y}_j$
- h or Δt: the 'step size' $t_j - t_{j-1}$ (if it is constant); otherwise h_j or Δt_j

Here we introduce **Euler's method**, and the framework to be used for better numerical methods later. We seek a numerical solution to the IVP

$$y' = f(t, y), \qquad y(a) = y_0$$

and suppose we wish to solve for $y(t)$ up to a time[1] $t = b$. The approximation will take the form of values $\tilde{y}_j$ defined on a grid

$$a = t_0 < t_1 < \cdots < t_N = b$$

such that

$$\tilde{y}_j \approx y(t_j).$$

For convenience, denote by y_j the **exact solution** at t_j and let the 'error' at each point be

$$e_j = y_j - \tilde{y}_j.$$

It will be assumed that we have a free choice of the t_j's. The situation is sketched in Figure 2.

3.1 Euler's method

Assume for simplicity that

$$h = t_j - t_{j-1},$$

the 'time step' size, is constant (not necessary, juts convenient).

Suppose that we have the exact value of $y(t)$. To get $y(t + h)$ from $y(t)$, expand in a Taylor series and use the ODE to simplify the derivatives:

$$\begin{aligned} y(t + h) &= y(t) + hy'(t) + O(h^2) \\ &= y(t) + hf(t, y) + O(h^2). \end{aligned}$$

[1] Here t is just the independent variable; time is a useful analogy since it suggests the direction (forward in time) in which the ODE is to be solved.

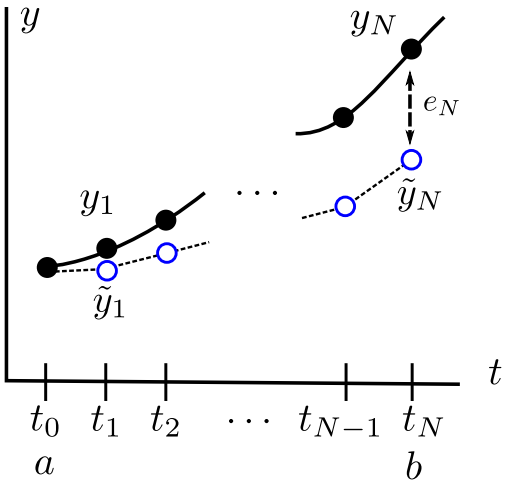


Figure 2: Numerical solution of an IVP forward in time from $t = a$ to $t = b$.

This gives, for any point t_j, the formula

$$y(t_j + h) = y(t_j) + hf(t_j, y(t_j)) + \tau_{j+1} \tag{3.1}$$

where τ_{j+1} is the **local truncation error** defined below. We could derive a formula, but the important thing is that

$$\tau_{j+1} = O(h^2).$$

Dropping the error in (3.1) and iterating this formula, we get **Euler's method**:

$$\widetilde{y}_{j+1} = \widetilde{y}_j + hf(t_j, \widetilde{y}_j). \tag{3.2}$$

The initial point is, ideally,

$$\widetilde{y}_0 = y_0$$

since the initial value (at t_0) is given. However, in practice, there may be some initial error in this quantity as well.

Notice that the total error is **not** just the sum of the truncation errors, because f is evaluated at the approximation $\widetilde{y}_n$. The truncation error will propagate through the iteration, as a careful analysis will show.

Definition (Local truncation error, or LTE): The **local truncation error** is the error incurred in obtaining $\tilde{y}_j$ when the previous data (y_{j-1} etc.) is known exactly. It is 'local' in the sense that it does not include errors created at previous steps.

Another interpretation is this: The local truncation error is what is 'left over' when the **exact solution** is plugged into the approximation. For Euler's method,

$$\widetilde{y}_{j+1} = \widetilde{y}_j + hf(t_j, \widetilde{y}_j)$$

if we plug in $y(t_j)$ instead of $\widetilde{y}_j$, the LHS/RHS are not equal; the difference is the local truncation error:

$$y_{j+1} = y_j + hf(t_j, y_j) + \tau_{j+1}.$$

3.2 Convergence

Suppose we use Euler's method to generate an approximation

$$(t_j, \widetilde{y}_j), \quad j = 0, \cdots, N$$

to the solution $y(t)$ in the interval $[a, b]$ (with $t_0 = a$ and $t_N = b$). The 'error' in the approximation that matters in practice is the **global error**

$$E = \max_{0 \le j \le N} |y_j - \widetilde{y}_j| = \max_{0 \le j \le N} |e_j|$$

where

$$e_j = y_j - \widetilde{y}_j$$

is the error at t_j. This is a measure of how well the approximation $\widetilde{y}_j$ agrees with the true solution over the whole interval.[2]

Our goal is to show that, given an interval $[a, b]$, the global error has the form

$$\max_{0 \le j \le N} |e_j| = O(h^p)$$

for some integer p, the order of the approximation. In this case we say that the method is **convergent**; the approximation, in theory, converges to the true solution $y(t)$ as $h \to 0$.[3]

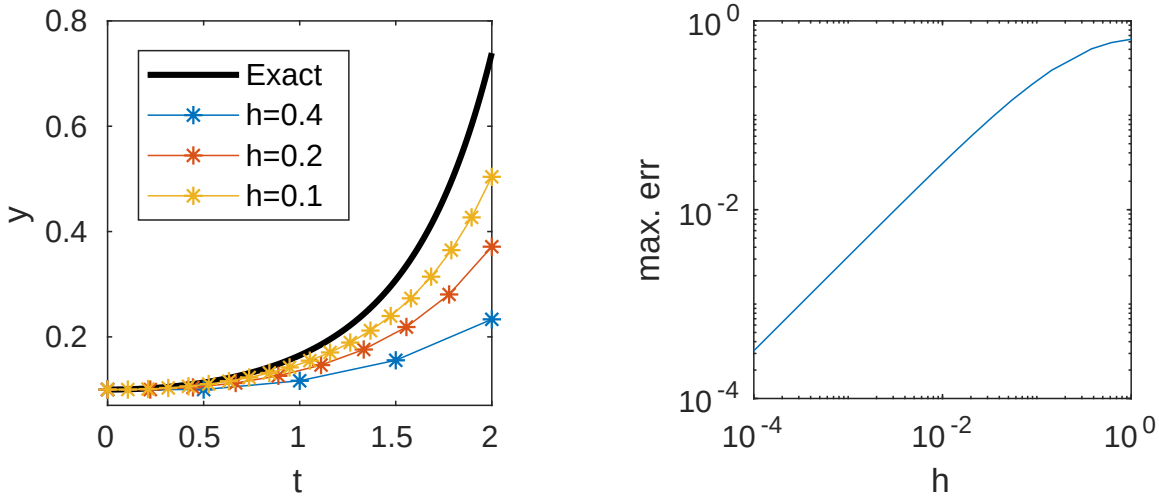
As an example, consider

$$y' = ty, \qquad y(0) = 0.1$$

which has exact solution $y(t) = 0.1e^{t^2}$. Below, we plot some approximations for various time-steps h; on the right is the max. error in the interval. The log-log plot has a slope of 1, indicating the error should be $O(h)$.

[2]Note that this is not precisely true since the approximation is not defined for all t; we would need to interpolate and that would have its own error bounds. But in practice we typically consider error at the points where the approximation is computed.

[3]See previous footnote; really this means that the approximation as a piecewise-defined function, e.g. piecewise linear, converges to $y(t)$ as $h \to 0$. Since the points get arbitrarily close together as $h \to 0$, the distinction between 'max error at the t_j's' and 'max error as functions' is not of much concern here.



3.3 Euler's method: error analysis

The details of the proof are instructive, as they will illustrate how error propagates and the 'worst case'. Assume that, as before, we have a fixed step size $h = t_j - t_{j-1}$, points

$$a = t_0 < t_1 < \cdots < t_N = b$$

and seek a solution in $[a, b]$. Further assume that

i) The partial derivative $\partial f/\partial y$ is bounded by L in the interval:

$$L := \max_{t\in[a,b], y\in\mathbb{R}} \left|\frac{\partial f}{\partial y}(t, y)\right| < \infty, \tag{3.3}$$

ii) The local truncation error τ_j is $O(h^2)$, uniform in j:

$$\max_{0\le j\le N} |\tau_j| = O(h^2).$$

This holds, for instance, if the appropriate derivatives of f are all bounded in $[a, b]$ (for all y), but we will not be too precise about the matter.

Condition (3.3) is called a **Lipschitz condition** and L is the **Lipschitz constant**. The relevant consequence is that

$$|f(t, y_1) - f(t, y_2)| \le L|y_1 - y_2| \text{ for all } t \in [a, b] \text{ and } y_1, y_2 \in \mathbb{R} \tag{3.4}$$

which is a direct consequence of the mean value theorem. This inequality is the key ingredient for the proof. The theorem is the following:

Theorem (Convergence for Euler's method): Let $\tilde{y}_j$ be the result of applying Euler's method (3.2) starting at $(t_0, \tilde{y}_0)$ and suppose that (i) and (ii) hold. Then

$$\max_{0\le j\le n} |e_j| \le e^{L(b-a)}|e_0| + \left(\frac{e^{L(b-a)}-1}{L}\right)\frac{\max|\tau_k|}{h}. \tag{3.5}$$

where τ_k is the local truncation error and $e_j = y_j - \tilde{y}_j$ is the error at t_j.

In particular, $\max \tau_j = O(h^2)$ and, as $h \to 0$, if $e_0 = 0$ (no error in y_0) then

$$\max_{0\le j\le n} |e_j| = O(h). \tag{3.6}$$

Proof. To start, recall that from the definition of the truncation error and the formula,

$$y_{j+1} = y_j + hf(t_j, y_j) + \tau_{j+1} \tag{3.7}$$
$$\widetilde{y}_{j+1} = \widetilde{y}_j + hf(t_j, \widetilde{y}_j) \tag{3.8}$$

Subtracting (3.8) from (3.7) gives

$$e_{j+1} = e_j + h\Big(f(t_j, y_j) - f(t_j, \widetilde{y}_j)\Big) + \tau_{j+1}.$$

From assumption (i) we have the Lipschitz property (3.4), so

$$|e_{j+1}| \le (1 + Lh)|e_j| + |\tau_{j+1}|.$$

Iterating, we get

$$|e_1| \le (1 + Lh)|e_0| + |\tau_1|,$$
$$|e_2| \le (1 + Lh)^2|e_0| + (1 + Lh)|\tau_1| + |\tau_2|$$

and in general

$$|e_j| \le (1 + Lh)^j|e_0| + \sum_{k=1}^{j}(1 + Lh)^{j-k}|\tau_k|.$$

Bounding each $|\tau_k|$ by the maximum in (ii) and evaluating the sum,

$$|e_j| \le (1 + Lh)^j|e_0| + (\max|\tau_k|)\,\frac{(1 + Lh)^j - 1}{Lh}.$$

Now we want to take the maximum over j, so the RHS must be written to be independent of j. To fix this problem, we use the crude estimate

$$(1 + Lh) \le e^{Lh}$$

to obtain

$$|e_j| \le e^{Ljh}|e_0| + \left(\frac{e^{Ljh} - 1}{Lh}\right)\max|\tau_k|$$

But $jh = t_j - t_0 \leq b - a$ (equal when $j = N$) so

$$|e_j| \leq e^{L(b-a)}|e_0| + \left(\frac{e^{L(b-a)} - 1}{L}\right) \frac{\max|\tau_k|}{h}.$$

Taking the maximum over j (note that the RHS is independent of j) we get (3.5).

Now if $\tilde{y}_0 = y_0$ i.e. the initial value is calculated exactly, then

$$\max_{0 \leq j \leq N} |e_j| \leq C \frac{\max|\tau_k|}{h}$$

for a constant C that depends on the interval size and L but not on h. By assumption (ii0, we know that $\max|\tau_k| = O(h^2)$ so the maximum error is $O(h)$. □

3.4 Interpreting the error bound

A few observations on what the error bound tells us:

- The LTE introduced at each step grows at a rate of e^{Lt} at worst.
- An initial error (in y_0) also propagates in the same way.
- The LTE is $O(h^2)$ and $O(1/h)$ steps are taken, so the total error is $O(h)$; the propagation does not affect the order of the error on a finite interval.

Note that the factor L and $(1 + Lh)$ are method dependent; for other methods, the factors may be other expressions related to L.

Our two examples from the theory,

$$\text{(a) } y' = ty, \qquad \text{(b) } y' = -ty$$

illustrate the propagation issue. As with actual solutions, the error and a numerical solution (or two nearby numerical solutions), can grow like e^{Lt} at worst. Indeed, for (a), the error grows in this way; the error bound is good here.

However, for (b), the numerical solutions actually converge to the true solution as t increases; in fact we have that the error behaves more like e^{-Lt}. But the error bound cannot distinguish between the two cases, so it is pessimistic for (b).

In either case, the global error over $[a, b]$ is $O(h)$ as $h \to 0$.

3.5 Order

The **order** p of a numerical method for an ODE is the order of the **global** truncation error. Euler's method, for instance, has order 1 since the global error is $O(h)$.

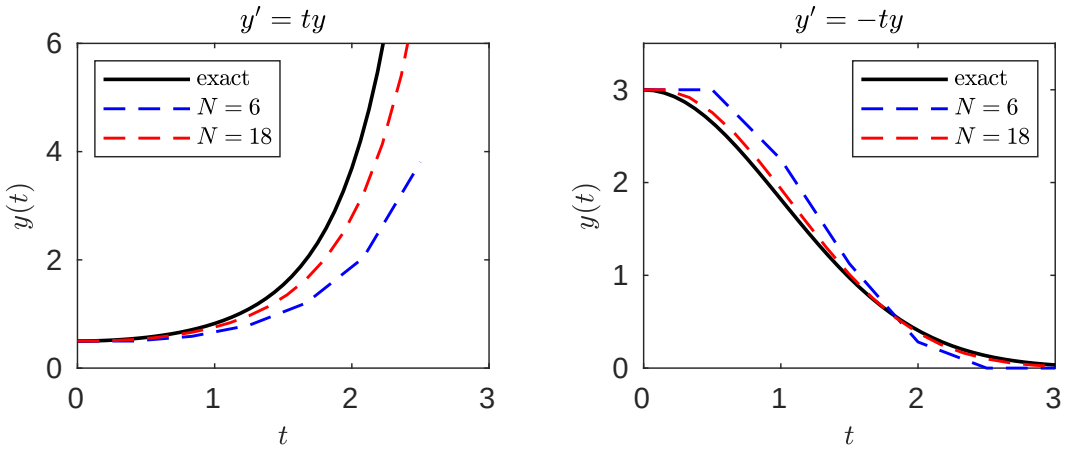


Figure 3: Numerical solutions to $y' = ty$ and $y' = -ty$ with different values of N; note the behavior of the error as t increases.

The $1/h$ factor is true for (most) other methods, so as a rule of thumb,

$$\text{LTE} = O(h^{p+1}) \implies \text{global error} = O(h^p).$$

The interpretation here is that to get from a to b we take $\sim 1/h$ steps, so the error is on the order of the number of steps times the error at each step, $(1/h) \cdot O(h^{p+1}) = O(h^p)$. The careful analysis shows that the order is not further worsened by the *propagation* of the errors.

Warning: Some texts define the LTE with an extra factor of h so that it lines up with the global error, in which case the rule is that the LTE and global error have the same order. For this reason it is safest to say that the error is $O(h^p)$ rather than to use the term 'order p', but either is fine in this class.

4 Consistency, stability, and convergence

Convergence is the property any good numerical method for an ODE must have. However, as the proof shows, it takes some effort (and is harder for more complicated methods). To better understand it, we must identify the key properties that guarantee convergence and how they can be controlled.

This strategy leads to two notions: **consistency** and **stability**, which are pieces that are necessary for convergence.

As before, we solve an IVP in an interval $[a, b]$ starting at $t_0 = a$ with step size h.

Definition (convergence): The numerical method is said to be **convergent** with order p if, given an interval $[a, b]$ where (i) as the timestep goes to zero:

$$\left(\max_{0 \le j \le n} |\widetilde{y}_j - y(t_j)| \right) = O(h^p) \text{ as } h \to 0.$$

4.1 Consistency and stability

A method is called **consistent** if

$$\text{the LTE at each step is } o(h) \text{ as } h \to 0.$$

That is,

$$\lim_{h \to 0} \frac{\tau_j}{h} = 0 \text{ for all } j.$$

To check consistency, we may assume **the result of the previous step is exact** (since this is how the LTE is defined). This is a benefit, as there is no need to worry about the accumulation of errors at earlier steps.

Example (Checking consistency): Euler's method,

$$\widetilde{y}_{j+1} = \widetilde{y}_j + hf(t_j, \widetilde{y}_j)$$

is consistent since the truncation error is

$$\tau_{j+1} = y(t_{j+1}) - y(t_j) - hf(t_j, y(t_j)) = \frac{h^2}{2} y''(\xi_j)$$

where ξ_j lies between t_j and t_{j+1}. For each j, the error is $O(h^2)$ as $h \to 0$. From the point of view of consistency, j is fixed so $y''(\xi_j)$ is just some constant; we do not need a uniform bound on y'' that is true for all j.

In constrast, **stability** refers to the sensitivity of solutions to initial conditions. First, it is worth reviewing the notion of stability for IVPs.

Stability for first-order IVPs: Consider the IVP

$$y' = f(t, y), \qquad y(t_0) = y_0 \tag{4.1}$$

and assume that

$$\left| \frac{\partial f}{\partial y}(t, y) \right| \le L \tag{4.2}$$

If (4.2) holds for some interval $t \in [a, b]$ containing t_0 and all y and f is continuous then the IVP (4.1) has a unique solution in $[a, b]$.
Moreoever, if y_1 amd y_2 are two solutions to the ODE with different initial conditions then

$$|y_1(t) - y_2(t)| \leq e^{L(t-t_0)}|y_1(t_0) - y_2(t_0)|.$$

We would like to have a corresponding notion of 'numerical' stability.

Definition (zero-stability) Suppose $\{y_n\}$ and $\{z_n\}$ are approximate solutions to (4.1) in $[a, b]$. If it holds that

$$|y_n - z_n| \leq C|y_0 - z_0| + O(h^p) \tag{4.3}$$

where C is *independent* of n, then the method is called **zero stable**.

Note that the best we can hope for is $C = e^{L(t-t_0)}$ since the numerical method will never be more stable than the actual IVP. In what follows, we will try to determine the right notions of stability for the numerical method.

As written, the stability condition is not easy to check. However, one can derive easy to verify conditions that imply zero stability. We have the following informal result:

Zero-stability, again: A 'typical' numerical method is **zero stable** in the sense (4.3) if it is numerically stable when used to solve the trivial ODE

$$y' = 0.$$

The condition is non-trivial to prove but much easier to check.

Here 'typical' includes any of the methods we consider in class (like Euler's method) and covers most methods for ODEs one encounters in practice.
With some effort, one can show that this notion of stability is exactly the minimum required for the method to converge.

Convergence theorem (Dahlquist) A 'typical' numerical method converges if it is **consistent** and **zero stable**. Moreover, it is true that

$$LTE = O(h^{p+1}) \implies \text{global error} = O(h^p).$$

This assertion was proven for Euler's method directly. Observe that the theorem lets us verify two simple conditions (easy to prove) to show that a method converges (hard to prove).

Example: convergence by theorem The Backward Euler method (to be studied in detail later) is

$$\widetilde{y}_{j+1} = \widetilde{y}_j + hf(t_{j+1}, \widetilde{y}_{j+1}).$$

Consistency: The local truncation error is defined by

$$y_{j+1} = y_j + hf(t_j + h, y_{j+1}) + \tau_{j+1}.$$

Expanding around $t_j + h$ and using the ODE we get

$$y_j = y_{j+1} - y'(t_{j+1})h + O(h^2) = y_{j+1} - hf(t_j + h, y_{j+1}) + O(h^2).$$

Plugging this in to the formula yields $\tau_{j+1} = O(h^2)$ so the method is consistent.

Zero stability: This part is trivial. When the ODE is $y' = 0$,

$$\widetilde{y}_{j+1} = \widetilde{y}_j$$

which is clearly numerically stable.

The theorem then guarantees that the method is convergent, and that the order of convergence is 1 (the global error is $O(h)$).

Note: It is mostly straightforward to prove convergence directly as we did for Euler's method; see HW for this.

4.2 Example of an unstable method

Obviously, any method we propose should be consistent (i.e. the truncation error is small enough). As the theorem asserts, consistency is not enough for convergence! A simple example illustrates what can go wrong when a method is consistent but not stable.

Euler's method can be derived by replacing y' in the ODE with a forward difference:

$$\frac{y(t+h) - y(t)}{h} = y' = f(t, y).$$

One might hope, then, that a more accurate method can be obtained by using a second-order forward difference

$$y'(t) = \frac{-y(t+2h) + 4y(t+h) - 3y(t)}{2h} + O(h^2).$$

Plugging this in, we obtain the method

$$-\widetilde{y}_{j+2} = -4\widetilde{y}_{j+1} + 3\widetilde{y}_j + 2hf(t_j, \widetilde{y}_j) \tag{4.4}$$

which is consistent with an $O(h^3)$ LTE. However, this method is not zero stable!

It suffices to show numerical instability for the trivial ODE $y' = 0$. The iteration reduces to

$$\widetilde{y}_{j+2} = 4\widetilde{y}_{j+1} - 3\widetilde{y}_j.$$

Plugging in $y_j = r^j$ we get a solution when

$$r^2 - 4r + 3 = 0 \implies r = 1,\, 3$$

so the general solution is

$$\widetilde{y}_j = a + b \cdot 3^j.$$

If initial values are chosen so that

$$\widetilde{y}_0 = \widetilde{y}_1$$

then $y_j = y_0$ for all j with exact arithmetic. However, if there are any errors ($\widetilde{y}_0 \neq \widetilde{y}_1$) then $b \neq 0$ and $|\widetilde{y}_j|$ willl grow exponentially. Thus, the method is unstable, and is not convergent.

Obtaining a second order method therefore requires a different approach.

5 Runge-Kutta methods

In this section the most popular general-purpose formulas are introduced, which can be constructed to be of any order.

5.1 Setup: one step methods

Euler's method is the simplest of the class of **one step** methods, which are methods that involve only y_j and intermediate quantities to compute the next value y_{j+1}. In contrast, **multi-step methods** use more than the previous point, e.g.

$$y_{j+1} = y_j + h(a_1 y_{j-1} + a_2 y_{j-2})$$

which will be addressed later.

Definition (one step methods): A general 'explicit' one step method has the form

$$\widetilde{y}_{j+1} = \widetilde{y}_j + h\psi(t_j, \widetilde{y}_j) \tag{5.1}$$

where ψ is some function we can evaluate at (t_j, y_j). The truncation error is defined by

$$y_{j+1} = y_j + h\psi(t_j, y_j) + \tau_{j+1}. \tag{5.2}$$

The term 'explicit' refers to the fact that y_{j+1} can be computed explicitly at each step, which makes computation easy.

To improve on the accuracy of Euler's method with a one-step method, we may try to include higher order terms to get a. To start, write (5.2) as

$$\underbrace{y_{j+1}}_{\text{LHS}} = \underbrace{y_j + h\psi(t_j, y_j)}_{\text{RHS}} + \tau_{j+1}$$

For a p-th order method, we want the LHS to equal the RHS up to $O(h^{p+1})$. Now expand the LHS in a Taylor series around t_j:

$$\text{LHS: } y_{j+1} = y(t_j) + hy'(t_j) + \frac{h^2}{2}y''(t_j) + \cdots$$

A p-order formula is therefore obtained by taking

$$\psi(t_j, y_j) = y'(t_j) + \frac{h}{2}y''(t_j) + \cdots + \frac{h^{p-1}}{p!}y^{(p-1)}(t_j).$$

The key point is that the derivatives of $y(t)$ at can be expressed in terms of f and its partial derivatives - which we presumably know. Simply differentiate the ODE $y' = f(t, y(t))$ in t, being **careful with the chain rule**. If $G(t, y)$ is any function of t and y evaluated on the solution $y(t)$ then

$$\frac{d}{dt}(G(t, y(t))) = G_t + G_y y'(t) = G_t + fG_y.$$

with subscripts denoting partial derivatives and G_t etc. evaluated at $(t, y(t))$.

It follows that

$$y'(t) = f(t, y(t)),$$
$$y''(t) = f_t + f_y f,$$
$$y'''(t) = (f_t + f_y f)' = f_{tt} + f_{ty}f + \cdots \text{ (see HW)}.$$

In operator form,

$$y^{(p)} = \left(\frac{\partial}{\partial t} + f\frac{\partial}{\partial y}\right)^{p-1} f.$$

Taylor's method: The p-th order one-step formula

$$y_{j+1} = y_j + hy'(t_j) + \frac{h^2}{2}y''(t_j) + \cdots + \frac{h^{p+1}}{(p+1)!}y^{(p+1)}(t_j) + O(h^{p+1}).$$

Note that $y', y'', \cdots$ are replaced by formulas involving f and its partials by repeatedly differentiating the ODE.

This method is generally not used due to the convenience of the (more or less equivalent) Runge-Kutta methods.

5.2 A better way

Taylor's method is inconvenient because it involves derivatives of f. Ideally, we want a one step method that needs to know $f(t, y)$ and nothing more.

The key observation is that the choice of ψ in Taylor's method is not unique. We can replace ψ with **anything else that has the same order error**. The idea of a **Runge-Kutta method** is to replace the expression with function evaluations at 'intermediate' points involving computable values starting with $f(t_j, y_j)$.

Let us illustrate this by deriving a second order one step method of the form

$$\underbrace{y_{j+1}}_{\text{LHS}} = \underbrace{y_j + w_1 h f_1 + w_2 h f_2}_{\text{RHS}} + O(h^3).$$

where

$$\begin{aligned} f_1 &= f(t_j, y_j), \\ f_2 &= f(t_j + h/2, y_j + h\beta f_1) \end{aligned}$$

and w_1, w_2, β are constants to be found.

Aside (integration): You may notice that this resembles an integration formula using two points; this is not a coincidence since

$$y' = f(t, y) \implies y_{j+1} - y_j = \int_{t_j}^{t_{j+1}} f(t, y(t))\, dt$$

so we are really estimating the integral of $f(t, y(t))$ using points at t_j and $t_{j+1/2}$. The problem is more complicated than just integrating $f(t)$ because the argument depends on the unknown $y(t)$, so that also has to be approximated.

To find the coefficients, expand everything in a Taylor series, keeping terms up to order h^2:

$$\begin{aligned} \text{LHS} &= y_j + h y_j' + \frac{h^2}{2} y_j'' + O(h^3) \\ &= y_j + hf + \frac{h^2}{2}(f_t + f f_y) + O(h^3) \end{aligned}$$

where f etc. are all evaluated at (t_j, y_j). For the f_i's, we only need to expand f_2:

$$\begin{aligned} h f_2 &= hf + \frac{h^2}{2} f_t + h^2 f_y(\beta f_1) + O(h^3) \\ &= hf + \frac{h^2}{2} f_t + \beta h^2 f f_y + O(h^3). \end{aligned}$$

Plugging this into the RHS gives

$$\begin{aligned}\text{RHS} &= y_j + h(w_1+w_2)f + \frac{w_2h^2}{2}f_t + w_2\beta h^2 f f_y + O(h^3)\\ \text{LHS} &= y_j + hf \qquad\qquad + \frac{h^2}{2}(f_t + f f_y) \qquad\quad + O(h^3)\end{aligned}$$

Comparing the LHS/RHS are equal up to $O(h^3)$ if

$$w_1 + w_2 = 1, \quad w_2 = 1, \quad w_2\beta = \frac{1}{2}$$

which gives

$$w_1 = 0, \quad w_2 = 1, \quad \beta = \frac{1}{2}.$$

We have therefore obtained the formula

$$y_{j+1} = y_j + hf_1 + hf_2 + O(h^3),$$

$$f_1 = f(t_j, y_j), \quad f_2 = f(t_j + h/2, y_j + hf_1/2),$$

which is called the **modified Euler method**.

Remark (integration connection): In this case one can interpret the formula as using the midpoint rule to estimate

$$y(t_{j+1}) = y(t_n) + \int_{t_j}^{t_{j+1}} f(t, y(t))\,dt \approx y(t_j) + hf(t_j + h/2, y(t_j + h/2))$$

but using Euler's method to estimate the midpoint value:

$$y(t_j + h/2) \approx y_n + \frac{h}{2}f(t_j, y_j).$$

In fact, when $f = f(t)$, the method reduces exactly to the composite midpoint rule for $\int f(t)\,dt$. However, in general, the various intermediate quantities do not have a clear interpretation, and the formulas can appear somewhat mysterious. Deriving methods from integration formulas is done in a different way (multistep methods).

5.3 Higher-order explicit methods

The modified Euler method belongs in the class of **Runge-Kutta** methods.

Definition (Explicit RK method) A general **explicit Runge-Kutta** (RK) method' uses m 'substeps' to get from y_j to y_{j+1} and has the form

$$\begin{aligned} f_1 &= f(t_j, y_j) \\ f_2 &= f(t_j + c_2 h, y_j + h a_{21} f_1) \\ f_3 &= f(t_j + c_3 h, y_j + h a_{31} f_1 + h a_{32} f_2) \\ &\vdots \\ f_m &= f(t_j + c_m h, y_j + h a_{m1} f_1 + \cdots + h a_{m,m-1} f_{m-1}) \\ y_{j+1} &= y_j + h(w_1 f_1 + \cdots + w_m f_m). \end{aligned}$$

Each f_j is an evaluation of f at a y-value obtained as y_j plus a linear combination of the previous f_i's. The 'next' value y_{j+1} is a linear combination of all the f_i's.

- The best possible local truncation error is $O(h^{p+1})$ where $p \leq m$. For each p, the system is underdetermined and has a family of solutions (see HW for the $p = 2$ case).
- Unfortunately, it is not true that $p = m$. That is, to get a high order method - fifth order and above - we need more substeps per iteration than the order.
- Deriving RK methods past third order is quite tedious and a mess of algebra, since the system for the coefficients is non-linear and the Taylor series expansions become complicated. (Exercise: verify that RK-4 below has an $O(h^5)$ LTE.)

Thankfully, just about every useful set of coefficients - at least for general purpose methods - has been calculated already, so in practice one can just look them up. They are typically arranged in the **Butcher Tableau** (see book for details).

The classical RK-4 method: One four step method of note is the classical 'RK-4' method

$$\begin{aligned} f_1 &= h f(t_n, y_n) \\ f_2 &= h f(t_n + \frac{1}{2}h, y_n + \frac{1}{2} f_1) \\ f_3 &= h f(t_n + \frac{1}{2}h, y_n + \frac{1}{2} f_2) \\ f_4 &= h f(t_n + h, y_n + f_3) \\ y_{n+1} &= y_n + \frac{1}{6}(f_1 + 2f_2 + 2f_3 + f_4). \end{aligned}$$

This method has a good balance of efficiency and accuracy (only four function evaluations per step, and $O(h^5)$ LTE). The method would be a good first choice for solving ODEs, except that there is a more popular variant that is better for error estimation.

5.4 Implicit methods

The explicit RK methods are nice because we simply 'iterate' to compute successive values, ending up with y_{j+1}. That is, all quantities in the formula can be computed explicitly.

However, we could also include y_{j+1} in the formula as well. For a one step method, the formula would take the form

$$y_{j+1} = y_j + \psi(t_j, y_j, y_{j+1}) + \tau_{j+1}.$$

Nothing is different in the theory; the truncation error is calculated in the same way and the same remarks on convergence apply. In practice, however, more work is required because y_{j+1} is defined implicitly in the formula:

$$\widetilde{y}_{j+1} = \widetilde{y}_j + \psi(t_j, \widetilde{y}_j, \widetilde{y}_{j+1}).$$

Suppose we have computed values up to $\widetilde{y}_j$. Define

$$g(z) = z - \widetilde{y}_j - \psi(t_{j+1}, \widetilde{y}_j, z).$$

Then $\widetilde{y}_{j+1}$ is a root of $g(z)$ (which is computable for any z). Thus $\widetilde{y}_{j+1}$ can be computed by applying Newton's method (ideally) or some other root-finder to $g(z)$.

Practical note: The obvious initial guess is $\widetilde{y}_j$, which is typically close to the root. If h is small, then $\widetilde{y}_j$ is almost guaranteed to be close to the root, and moreover

$$\widetilde{y}_{j+1} \to \widetilde{y}_j \text{ as } h \to 0.$$

Thus, if Newton's method fails to converge, h can be reduced to make it work. Since the initial guess is close, quadratic convergence ensures that the Newton iteration will only take a few steps to achieve very high accuracy - so each step is only a few times more work than an equally accurate explicit method.

You may wonder why we would bother with an implicit method when the explicit methods are more efficient per step; the reason is that they have other desirable properties to be explored in the next section. For some problems, implicit methods can use much larger h values than explicit ones.

5.4.1 Example: using Backward Euler

Suppose we wish to solve

$$y' = -t \sin y, \quad y(0) = y_0$$

using the Backward Euler method

$$y_{j+1} = y_j + hf(t_{j+1}, y_{j+1}).$$

If Newton's method is used, the code must know f and $\partial f/\partial y$. The function in Matlab may be written, for instance, in the form

```
[T,Y] = beuler(f,fy,[a b],y_0,h)
```

where `f,fy` are both functions of t and y. Internally, at step j we have

$$g(z) = z - y_j - hf(t_{j+1}, z), \quad g'(z) = 1 - hf_y(t_{j+1}, z)$$

and the Newton iteration is

$$z_{k+1} = z_k - \frac{g(z_k)}{g'(z_k)}.$$

This is iterated until convergence; then y_{j+1} is set to the resulting z.

5.4.2 Example: deriving the trapezoidal method

Here we derive a second-order method that uses $f(t_j, y_j)$ and $f(t_{j+1}, y_{j+1})$. The formula is

$$y_{j+1} = y_j + hw_1 f(t_j, y_j) + hw_2 f(t_{j+1}, y_{j+1}) + \tau_{j+1}.$$

First, note that for the RHS, we only need to expand y_{j+1} up to an $O(h^2)$ error. Using

$$y_{j+1} = y_j + hf + O(h^2),$$

we have that

$$f(t_{j+1}, y_{j+1}) = f(t_j + h, y_j + A)$$

where $A = hf + O(h^2)$. Since $A^2 = O(h^2)$, the result is

$$\begin{aligned}
\text{RHS} &= hw_1 f + hw_2 f(t_j + h, y_{j+1}) \\
&= hw_1 f + hw_2 \Big(f + hf_t + f_y A + O(A^2) \Big) \\
&= h(w_1 + w_2) f + w_2 h^2 \Big(f_t + f f_y \Big) + O(h^3).
\end{aligned}$$

Comparing to the LHS,

$$\text{LHS} = y_{j+1} = y_j + hf + \frac{h^2}{2}(f_t + ffy) + O(h^3)$$

we find that $w_1 = w_2 = 1/2$.

Trapezoidal method: The implicit formula

$$y_{j+1} = y_j + \frac{h}{2}(f_j + f_{j+1}) + O(h^3)$$

where $f_j = f(t_j, y_j)$ and $f_{j+1} = f(t_{j+1}, y_{j+1})$.

Note that when $f = f(t)$, the formula reduces to the composite trapezoidal rule.