An Introduction to Numerical Methods for ODEs

Mariya Savinov, email: mariyasavinov@nyu.edu

A few remarks:
These lecture notes are intended to be roughly two lectures of material, providing an introduction to solving *ordinary differential equations* (ODEs) numerically. These notes have primarily been adapted from:

- *Elementary Numerical Analysis* by Atkinson and Han, 3rd edition, Chapter 8
- *Elementary Differential Equations and Boundary Value Problems* by Boyce and DiPrima, Chapter 8
- *Numerical methods for ordinary differential equations - initial value problems* by Griffiths and Higham, Chapters 1 and 2

For those who have not taken an ODEs course, Boyce and DiPrima's text provides an introduction to the theory and applications at the undergraduate level. However, these notes are aimed at students who have taken multivariable calculus – no prior knowledge of ODEs is required, as we will give a brief introduction to first order differential equations to start.

1 Introduction

In science, engineering, economics, sociology, and a number of other disciplines, it is often necessary to understand the *rate at which things change*. Ordinary differential equations (ODEs) provide a way to describe and predict the nature of chemical reaction systems, n-body interactions in physics, current flow in electronic circuits, long-term economic growth, population statistics, and more. The majority of ODEs, however, do not have closed-form, known solutions. As such, it is necessary to have methods which allow us to approximate the true, unknown solutions of ODEs in a way which is robust with reliable accuracy. Over the past 50 or so years of scientific computing, a great number of advancements, theoretical and practical, have made it that robustness and accuracy are attainable for many ODEs of interest. Nonetheless, many challenges remain, which we will not even begin to address here.

In these lecture notes, we provide an introduction to numerical methods for ODEs. We will introduce first-order differential equations, providing examples with and without solutions, as well as a statement on existence and uniqueness. We then discuss the Euler methods, motivating us to define error and convergence. Given an understanding of the rate of convergence, Richardson extrapolation will be introduced as a way to improve accuracy. Finally, we present two second order methods, ending with introductions to Runge-Kutta Methods and Multistep methods.

1.1 First-order differential equations

We will limit our focus to *first-order differential equations*, in particular to *initial value problems* of the form

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{1}$$

For some simple ODEs, it is possible to find closed form solutions. Let us illustrate a simple case where f depends only on time t:

Example 1.1.1. Consider the problem where the position y of, e.g., a particle, changes in time t like a sinusoidal:

$$y'(t) = \sin(t)$$

Integrating both sides, it is clear that the general solution is

$$y(t) = \sin(t) + c$$

where c is an arbitrary integration constant. Without any additional information, there are infinitely-many solutions to the problem $y'(t) = \sin(t)$. However, suppose that we also know that at initial time $t = 0$, the position of the particle is $y(0) = 3$. Then we have a *unique* solution for $t \geq 0$:

$$y(t) = \sin(t) + 3$$

As illustrated above, the general solution of first-order ODEs depends on an arbitrary integration constant that must be specified with an additional condition. This condition appears in (1) as $y(t_0) = y_0$, an "initial value". However, setting an initial condition does not necessarily guarantee the existence of a unique solution. Depending on the properties of $f(t, y)$ and choice of initial condition y_0, one may only have a unique solution on a finite interval $t \in [t_0, T]$, or not at all.

Example 1.1.2. For example, consider the initial value problem:

$$\begin{cases} y'(t) = 2t\,[y(t)]^2 \\ y(0) = y_0 \end{cases}$$

The ODE is *separable* and can easily be solved analytically, yielding the following unique solution to the IVP:

$$y(t) = \frac{1}{-t^2 + y_0^{-1}}$$

Notice that, depending on the sign of y_0, the solution may or may not exist for all $t \geq 0$. If $y_0 < 0$, then the solution exists $\forall t \geq 0$. However, if $y_0 > 0$, then the solution does not exist when $t = \sqrt{1/y_0}$. I.e., the solution exists only on the finite interval $t \in \left[0, \sqrt{1/y_0}\right)$.

1.2 Existence and Uniqueness of the IVP

There exists a general theorem on the existence of a unique solution to the IVP (1), which we will not prove but simply state here:

Theorem 1.2.1. Let $f(t, y)$ and $f_y(t, y)$ be continuous functions of t and y at all points (t, y) in some neighborhood S of (t_0, y_0) (see Fig. 1). Then, there is a unique function $y = \phi(t)$ defined on some interval $t \in [t_0 - \delta, t_0 + \delta]$ satisfying (1).

Remark 1.2.1. Theorem 1.2.1 only guarantees existence and uniqueness in an interval close to the initial condition – this interval itself might be quite small. Additionally, the continuity of $f(t, y)$ and $f_y(t, y)$ do not imply the existence of a $\phi(t)$ which is continuous for the same t.

Example 1.2.1. Let us return to Example 1.1.2. In this case, we had $f(t, y) = -2ty^2$, thus $f_y(t, y) = 4ty$. Both of these functions are continuous for all (t, y) pairs. However, the solution only exists on the interval $t \in \left[-\sqrt{1/y_0}, \sqrt{1/y_0}\right]$. If the initial condition large, $y_0 \gg 1$, then this interval of existence is quite small.

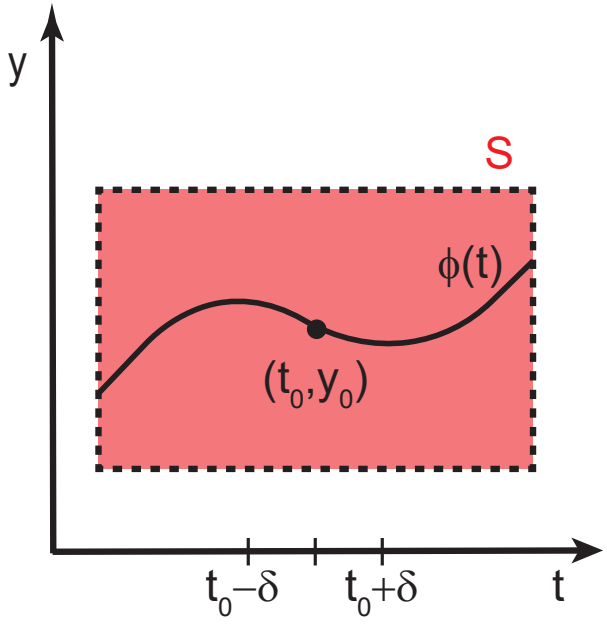

Figure 1: Domain S with a unique solution $\phi(t)$ from Thm. 1.2.1.

1.3 The need for numerical methods

In Examples 1.1.1 and 1.1.2, we had analytical solutions, so a numerical method would not be necessary. However, the general first-order initial value problem (1) may or may not be possible to solve analytically.

Example 1.3.1. For example, suppose you have

$$y'(t) = \exp\left(-t\, y(t)^4\right)$$

Regardless of choice of initial condition, this ODE cannot be solved analytically. So, we must resort to numerical methods.

2 Basic numerical methods for solving the IVP

We will concentrate on the first-order IVP (1), from now on assuming f and f_y are continuous on some region S in the (t, y) plane containing (t_0, y_0). Then by Theorem 1.2.1, there exists a unique solution $y = \phi(t)$ in some interval about t_0. We will further assume that this $\phi(t)$ solution exists in our interval of interest for all remaining problems.

Suppose we want to approximate the solution $\phi(t)$ of (1) in some interval $t \in [t_0, T]$. Of course, we cannot solve $\forall t$ in this interval, so we must consider discrete nodes $t_0 < t_1 < ... < t_{n-1} < t_n \leq T$. Our aim is to construct $y_1, y_2, ..., y_n$ approximate values of the true solution $\phi(t)$ at these $t_1, t_2, ..., t_n$ times. For simplicity, we assume that the times t_i are equally spaced, such that the timestep $t_{i+1} - t_i = h$ for all i indices. Though these methods do not all require equally space timesteps, this simplification eases discussions of *convergence* and *stability*.

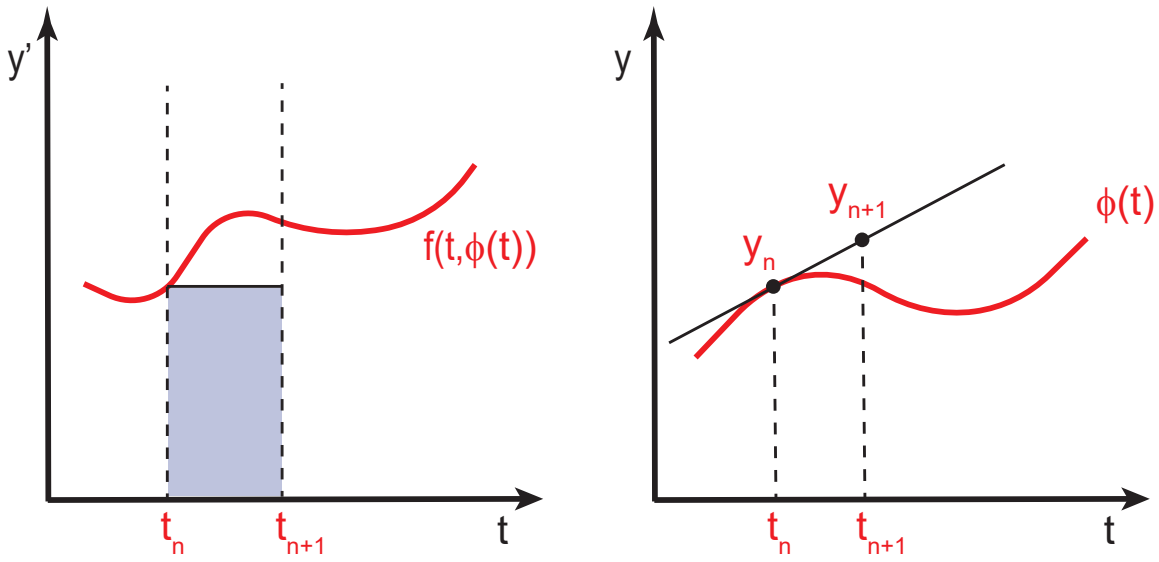

Figure 2: (Left) Approximation of the integral in (2) to derive the Euler Method. (Right) Graphical depiction of one step of the Euler method (3).

The methods we mention here can be derived in a number of ways. We choose here to approach numerical methods for IVPs from the perspective of integral equations, and how we can approximate integrals (numerical integration). Notice that the IVP (1) can be rewritten as an integral equation: given a solution $y = \phi(t)$, we have

$$\frac{d\phi}{dt}(t) = f\left(t, \phi\left(t\right)\right)$$

At some point $t = t_n$, this is equivalent to the integral equation:

$$\int_{t_n}^{t_{n+1}} \phi'(t)\, dt = \int_{t_n}^{t_{n+1}} f(t, \phi(t))\, dt$$

Integrating both sides, it follows that

$$\phi\left(t_{n+1}\right) = \phi\left(t_n\right) + \int_{t_n}^{t_{n+1}} f\left(t, \phi(t)\right)\, dt \tag{2}$$

The integral in the above right-hand-side defines the area under the curve $f\left(t, \phi(t)\right)$ from t_n to t_{n+1}. Intuitively, if we can approximate this integral "accurately," it should lead us to an "accurate" numerical method (we will later discuss what we mean by "accurate").

2.1 Euler / Tangent Line Method

One simple approach to the integral in (2) is to approximate $f\left(t, \phi(t)\right)$ by a constant – say, $f\left(t_n, \phi\left(t_n\right)\right)$, as shown in the left plot of Fig. 2. Then, (2) becomes:

$$\phi\left(t_{n+1}\right) \approx \phi\left(t_n\right) + \left(t_{n+1} - t_n\right) f\left(t_n \phi\left(t_n\right)\right)$$

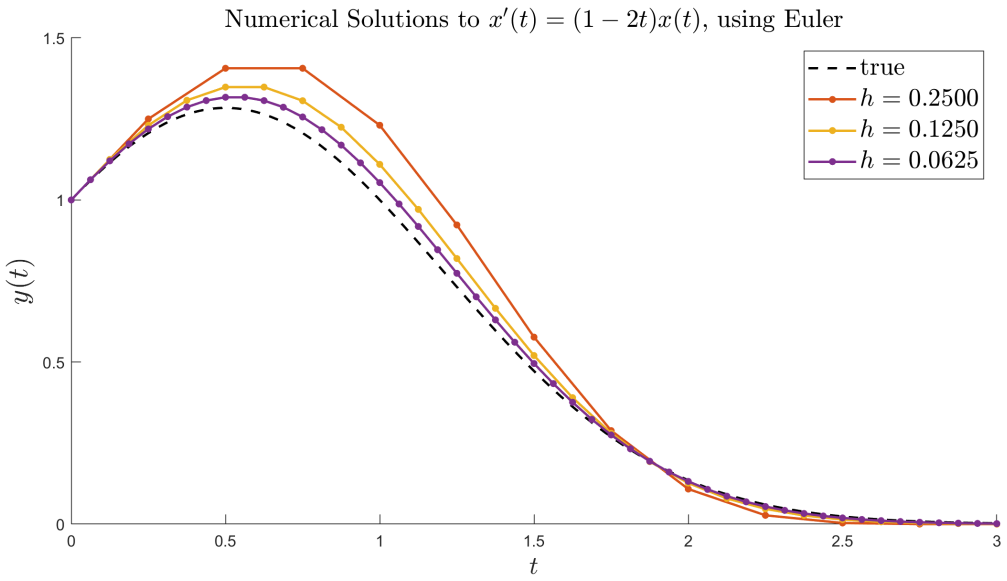

Figure 3: Approximate solutions to the IVP in Example 2.1.1, with true soloution shown.

Replacing $\phi(t_n)$ with its approximate value y_n, following our earlier notation, we arrive at the *Euler method*:

$$y_{n+1} = y_n + h\, f(t_n, y_n) \tag{3}$$

Geometrically, one can think of this as stepping forward along the tangent line to the graph $y = \phi(t)$ at time t_n (the slope at this point is $\phi'(t_n) = f(t_n, \phi(t_n))$), as shown in right plot of Fig. 2.

Example 2.1.1. Consider the case of the IVP

$$\begin{cases} y'(t) = (1-2t)y(t) \\ y(0) = 1 \end{cases}$$

on the interval $0 \le t \le 3$. We choose a case with a true solution:

$$y(t) = \exp\left(\frac{1}{4} - \left(\frac{1}{2} - t\right)^2\right)$$

so we can consider the error of the method. In this case, one step of the Euler method (3) is

$$y_{n+1} = y_n + h\,(1-2t_n)\,y_n = (1 + h\,(1-2t_n))\,y_n$$

Suppose we consider timesteps $h = 0.25,\ 0.125,\ 0.0625$ (each timestep is half the previous). Fig. 3 shows the numerical approximation in time, plotted against the true solution. It is clear that the smaller timestep h you consider, the better the approximation. Considering even smaller timesteps, we can look directly at the maximum deviation from the true solution in all time (L_∞ error), finding that:

h	Euler L_∞ error
0.25	0.23047
0.125	0.10967
0.0625	0.05405
0.03125	0.02674
0.015625	0.013308

Notice how considering half the timestep tends to half the error. We will return to this example when discussing error and convergence rate.

Remark 2.1.1. One alternative way to derive the Euler method is to rewrite $\phi'(t)$ with a forward difference quotient, giving

$$\frac{\phi(t_{n+1}) - \phi(t_n)}{t_{n+1} - t_n} \approx f(t_n, \phi(t_n))$$

Replacing $\phi(t_n)$ with its approximate value y_n, you once again arrive at (3).

Another perspective is to approximate $\phi(t_n)$ with its Taylor series about t_n:

$$\begin{aligned}\phi(t_n + h) &= \phi(t_n) + \phi'(t_n)\, h + \phi''(t_n)\, \frac{h^2}{2} + ... \\ &= \phi(t_n) + f(t_n, \phi(t_n))\, h + \phi''(t_n)\, \frac{h^2}{2} + ...\end{aligned}$$

Keeping only the zeroth and first order terms (in h), you once again have Euler's method. Notice a more accurate formula can be obtained if more terms in the series are retained (this is the idea behind Taylor series methods for timestepping), but this requires taking higher derivatives of $f(t, y)$, and the formula quickly becomes complicated. It is also possible to estimate the magnitude of the error by considering the Taylor series remainder term, which we will return to shortly.

2.2 Backward Euler

For Euler, we approximated the integral equation (2) by setting $f(t, \phi(t)) \approx f(t_n, \phi(t_n))$. If instead we approximate $f(t, \phi(t))$ with the value at time t_{n+1}, as shown on the left in Fig. 4, we arrive at the *Backward Euler* method:

$$y_{n+1} = y_n + h\, f(t_{n+1}, y_{n+1}) \tag{4}$$

Notice however that this equation defines y_{n+1} *implicitly*.

Remark 2.2.1. Depending on the nature of f, it may not be easy to solve this equation to get y_{n+1} at the next time. A possible approach in such cases is to use an iteration technique:

$$y_{n+1}^{(j+1)} = y_n + hf\left(t_{n+1}, y_{n+1}^{(j)}\right)$$

So long as h is sufficiently small, conditionally dependent on the nature of f_y, as $j \to \infty$ then $y_{n+1}^{(j+1)}$ will converge to y_{n+1}. In practice, it is often sufficient to do this iteration only once – however, the stability of the method will be different.

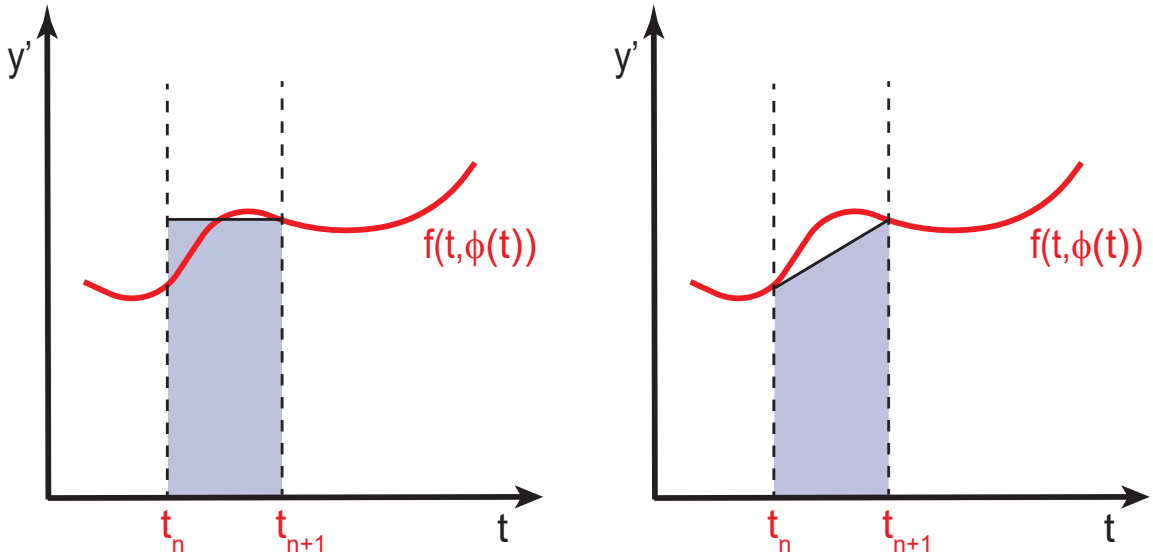

Figure 4: Approximation of the integral in (2) to derive the Backward Euler Method (Left) and Trapezoidal Method (Right).

Example 2.2.1. Let us return to the IVP we visited in Example 2.1.1. In this case, we can actually rewrite the implicit equation to have an explicit formula for each timestep:

$$\begin{aligned} y_{n+1} &= y_n + h\left(1 - 2t_{n+1}\right) y_{n+1} \\ y_{n+1}\left(1 - h\left(1 - 2t_{n+1}\right)\right) &= y_n \\ \implies y_{n+1} &= \frac{1}{1 - h\left(1 - 2t_{n+1}\right)} y_n \end{aligned}$$

We can then consider the maximum error over the time domain $t \in [0, 3]$ for Backward Euler, and compare this to our results with the Euler method:

h	Euler	Backward Euler
0.25	0.23047	0.19036
0.125	0.10967	0.10177
0.0625	0.05405	0.051833
0.03125	0.02674	0.026218
0.015625	0.013308	0.013174

Notice that Backward Euler appears approximately as accurate *for this case* as Euler when h is small, and the error follows a similar trend with decreasing h. This once again prompts us to consider defining *convergence* and *error*, and how these properties depend on our numerical method.

3 Convergence and Error

Thus far, we have not considered the issue of *convergence*: for a chosen numerical method, is it true that

$$\lim_{\substack{h\to 0 \\ nh=T-t_0}} y_n = \phi(T)$$

Meaning, as you take smaller timesteps h (increasing n proportionately such that y_n is the approximation of ϕ at time T), does y_n converge to the true value? Moreover, if so, how quickly does the approximation over the interval converge? How small of a timestep h does one need to achieve a desired accuracy? To answer these questions, we need to qualify *error* and understand the relationship error and a chosen step-size h.

3.1 Global and local truncation error

We define the *global truncation error* as

$$E_n = \phi(t_n) - y_n$$

i.e. the error due to applying an *approximate formula* to *approximate data*, since inherently y_n depends on the previous steps $y_{n-1}, y_{n-2}, \ldots$. Suppose however that we know the previous steps *exactly*, computing y_n^* with known $\phi(t)$ at previous times. Then, we define the *local truncation error* as the error induced by *one step* of an approximate formula to *exact data*:

$$e_n = \phi(t_n) - y_n^*$$

For example, if we are interested in the local truncation of Euler (3), y_n^* would be given by

$$y_n^* = \phi(t_{n-1}) + h\, f(t_{n-1}, \phi(t_{n-1}))$$

So e_n is the error introduced by taking *one* approximate step.

In reality, computations also suffer from *round-off error*, since computers only have finite precision. However, as a general rule, until h is very small, the global truncation error will dominate the observed error.

3.2 Local truncation error of the Euler method

In the case of the Euler method, we will now demonstrate how one can derive the local truncation error of a numerical method. Let us assume that the true solution $\phi(t)$ of the IVP (1) has a continuous 2nd derivative (note that this is true if f, f_y, f_t are continuous). Then, expand $\phi(t)$ about t_n with a Taylor polynomial:

$$\begin{aligned}\phi(t_n + h) &= \phi(t_n) + h\phi'(t_n) + \frac{1}{2}\phi''(\xi_n)\, h^2 \\ &= \phi(t_n) + hf(t_n, \phi(t_n)) + \frac{1}{2}\phi''(\xi_n)\, h^2\end{aligned}$$

where for a given h, there exists some $\xi_n \in [t_n, t_n + h]$.

Now, consider y^*_{n+1} be the approximation of $\phi(t_{n+1}) = \phi(t_n + h)$ using one step of Euler with *exact* data, i.e.

$$y^*_{n+1} = \phi(t_n) + hf(t_n, \phi(t_n))$$

It follows that the local truncation error at $t_{n+1} = t_n + h$ is

$$e_{n+1} = \phi(t_n + h) - y^*_{n+1} = \frac{1}{2}\phi''(\xi_n)\, h^2 \tag{5}$$

It is natural then that, in general, the local truncation error on the interval of interest $[t_0, T]$ is bounded as:

$$|e_n| \leq \frac{1}{2}Mh^2 \qquad \text{where } M = \max_{[t_0,T]} |\phi''(t)| \tag{6}$$

Note that this bound is typically an *overestimation*. One might be tempted to use (5) and (6) to estimate the error of the numerical solution – however, estimating M may not be easy or possible. The most important point here is that (6) is a statement about the *order of convergence* of the local truncation error.

3.3 Orders of convergence

We say that the local truncation error converges with order p in h if it satisfies the inequality:

$$|e_n| \leq Ch^p \implies \text{equivalently, } |e_n| = O(h^p) \tag{7}$$

Note that this means that as $h \to 0$, $|e_n| \to 0$ at least as fast as $h^p \to 0$. So, in the previous section, we demonstrated that the local truncation error of the Euler method converges with second order. The larger the order p, the faster convergence you expect.

Our interest, however, is often in *global* accuracy. While one can derive the global order of convergence for a method, the local truncation error tends to be more easily accessible and is an easy measure of the accuracy of a numerical method. As a general rule-of-thumb, the order of convergence for the global truncation error is typically *one order lower* than the local truncation error. So if you have a method of order p, this means, given our definitions of local and global truncation error,

$$|e_n| = O\left(h^{p+1}\right) \quad \text{and} \quad |E_n| = O(h^p) \tag{8}$$

Note that this considers the global error *on a finite interval*. A technically wrong but intuitive explanation is as follows: suppose at each step $i = 1, 2, ..., n$ you make an error that is approximately the size of your local truncation error $\sim e_i$. As you take more steps, this error accumulates, such that at n steps you have global error like

$$E_n \sim \sum_{i=1}^{n} e_i$$

If the method has local truncation error which is $O(h^{p+1})$, then

$$|E_n| \leq \sum_{i=1}^{n} |e_i| \leq \sum_{i=1}^{n} Ch^{p+1} = Cnh^{p+1}$$

Necessarily, n relates to the step-size such that $n = (t_f - t_0)/h$. So,

$$|E_n| \leq C(t_f - t_0) h^p = \tilde{C} h^p$$

Thus, the global truncation error is $O(h^p)$, one order lower than the local truncation error.

However, note that error is not strictly additive in this way – the local truncation error e_i at step i may be compounded by errors in later steps, or there may be a dependence on the nature of the ODE. This explanation is especially not true if the method is not *stable*. In the next section, we will give a more rigorous proof for Euler in the case of linear f (the proof for the general case is rather technical, because f may be nonlinear, so we will illustrate the idea through Euler in an easier, linear case).

We can now put our observations of the behavior of the Euler and Backward Euler methods in the case given by Example 2.1.1 into perspective: since Euler and Backward Euler are *first order* methods (the global truncation error decays like $O(h)$), halving the timestep h results in roughly halving the error (it is not exactly, since the order of convergence is true only *asymptotically*).

3.4 Global truncation error of Euler

To illustrate how global truncation error can be derived from the local truncation error, let us consider the linear IVP:

$$\begin{cases} y'(t) = \lambda y(t) + g(t) \\ y(t_0) = y_0 \end{cases}$$

Applying Euler's method to this problem yields steps like

$$y_{n+1} = y_n + h(\lambda y_n + g(t_n)) = (1 + \lambda h) y_n + h g(t_n) \tag{9}$$

Recall that in deriving the local truncation error e_{n+1} of the Euler method, we considered a Taylor expansion of $\phi(t_{n+1})$. Doing the same for this case, we get

$$\phi(t_{n+1}) = \phi(t_n) + h\phi'(t_n) + \frac{h^2}{2}\phi''(\xi_n) = (1 + \lambda h)\phi(t_n) + h g(t_n) + e_{n+1} \tag{10}$$

where we have replaced the Taylor remainder with e_{n+1}, as it is identically the local truncation error. Subtracting (9) from (10), we have

$$\phi(t_{n+1}) - y_{n+1} = (1 + \lambda h)(\phi(t_n) - y_n) + e_{n+1}$$

Notice that this gives a relation between the global truncation error at t_{n+1} and t_n:

$$E_{n+1} = (1 + \lambda h) E_n + e_{n+1}$$

The first few global truncation errors are then

$$\begin{aligned} E_1 &= e_1 \\ E_2 &= (1 + \lambda h) E_1 + e_2 = (1 + \lambda h) e_1 + e_2 \\ E_3 &= (1 + \lambda h) E_2 + e_3 = (1 + \lambda h)^2 e_1 + (1 + \lambda h) e_2 + e_3 \\ &\vdots \\ E_n &= \sum_{j=1}^{n} (1 + \lambda h)^{n-j} e_j \end{aligned}$$

From here, we can proceed with bounding the global truncation error E_n. We know that

$$|1 + \lambda h| \leq \exp(h|\lambda|)$$

so

$$|1 + \lambda h|^{n-j} \leq \exp((n-j)h|\lambda|) \leq \exp(nh|\lambda|)$$

Note that $nh = t_f - t_0$. Thus, knowing the local truncation error is $O(h^2)$, we have

$$\begin{aligned}
|E_n| &\leq \sum_{j=1}^{n} |1 + \lambda h|^{n-j} |e_j| \\
&\leq \sum_{j=1}^{n} \exp((t_f - t_0)|\lambda|) Ch^2 \\
&= n \exp((t_f - t_0)|\lambda|) Ch^2 \\
&= (t_f - t_0) \exp((t_f - t_0)|\lambda|) Ch
\end{aligned}$$

Therefore, we have that

$$|E_n| \leq \tilde{C}h$$

where $\tilde{C} = (t_f - t_0) \exp((t_f - t_0)|\lambda|) C$. This is exactly saying that the global truncation error of Euler is $O(h)$.

3.5 Richardson Extrapolation

Considering the behavior of the leading order term in the global error over the domain $[t_0, T]$ requires knowledge of the true solution $\phi(t)$ of (1), which we rarely have. However, if we know the order of convergence of the method, we can improve the accuracy of our numerical solution through a method called *Richardson Extrapolation*.

Suppose we have two numerical solutions, y_h and $y_{h/2}$, at time t^* using timestep sizes h and $h/2$, respectively. Additionally, suppose the numerical method used has order p. Let $\phi(t^*)$ denote the true solution. Then, we know that

$$\begin{aligned}
\phi(t^*) - y_h &= C h^p + O\left(h^{p+1}\right) \\
\phi(t^*) - y_{h/2} &= C \left(\frac{h}{2}\right)^p + O\left(h^{p+1}\right)
\end{aligned}$$

Note C is *the same* constant since we are considering error at time t^*. Subtracting the two, we have

$$y_h - y_{h/2} + \left[C h^p \left(1 - \frac{1}{2^p}\right)\right] = O\left(h^{p+1}\right)$$

Rearranging,

$$C h^p = -\frac{2^p}{2^p - 1}\left(y_h - y_{h/2}\right) + O\left(h^{p+1}\right)$$

Thus, the first term on the right-hand-side is an estimate of the leading-order term of the error for y_h. Specifically,

$$-\frac{2^p}{2^p - 1}\left(y_h - y_{h/2}\right)$$

is the *Richardson error estimate* of y_h. We can then construct a *better* numerical solution which we call the *Richardson Extrapolation* of the solution:

$$R(h, t^*) = y_h + Ch^p = y_h - \frac{2^p}{2^p - 1}\left(y_h - y_{h/2}\right) = \frac{2^p\, y_{h/2} - y_h}{2^p - 1} \tag{11}$$

Notice, necessarily by our derivation, that

$$\phi(t^*) - R(h, t^*) = O\left(h^{p+1}\right)$$

Therefore, via Richardson extrapolation we increased the order of convergence from p to $p+1$.

Remark 3.5.1. "Tricks" like those above used to estimate the error and increase accuracy are employed in adaptive timestepping methods!

4 More methods for solving IVPs

So far, the numerical methods we have considered are the Euler and Backward Euler methods. Both are *1st order* methods, whose global truncation error decays like $O(h)$. Due to their low order of convergence, they are rarely used in practice, but are easy methods which help demonstrate fundamental principles of numerical methods for IVPs.

Note that the larger the desired order p, the more "complicated" and/or expensive computationally the methods become to solve for y_{n+1}. On the other hand, the smaller your timestep h, the more timesteps are necessary to approximate the solution on a finite time interval; moreover, if h becomes too small, or too many steps are required in a calculation, roundoff error may begin to dominate. So a numerical method should be chosen wisely to balance these factors, and others.

Recall that we derived the Euler and Backward Euler methods through crudely approximating the integral in (2) by replacing $f(t, \phi(t))$ with a constant defined either by its value at t_n (Euler) or t_{n+1} (Backward Euler). We can obtain better, higher order, methods by approximating this integral more accurately. Here we outline a few 2nd order methods which do so.

4.1 Trapezoidal Method

Suppose we approximate the integral (2) using the trapezoidal rule – i.e., approximate the area under the curve with a trapezoid defined by the initial value at t_n and final value at t_{n+1}, as shown in the right of Fig. 4. Then we have

$$\int_{t_n}^{t_{n+1}} f(t, \phi(t))\, dt \approx (t_{n+1} - t_n)\, \frac{f(t_n, \phi(t_n)) + f(t_{n+1}, \phi(t_{n+1}))}{2}$$

This then leads to the *Trapezoidal Method* for timestepping the IVP:

$$y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right) \tag{12}$$

Notice that this method can also be obtained by taking the average of the Euler and Backward Euler methods. The trapezoidal method is, unsurprisingly, second order accurate, but note that (12) still defines the next step y_{n+1} implicitly.

Example 4.1.1. Returning to the IVP we visited in Example 2.1.1, this is, similarly to Backward Euler, also a case where we can write an explicit solution to the implicit equation at each timestep.

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{2}\left(\left(1-2t_n\right)y_n + \left(1-2t_{n+1}\right)y_{n+1}\right) \\ y_{n+1}\left(1-h\left(1-2t_{n+1}\right)/2\right) &= \left(1+h\left(1-2t_n\right)/2\right)y_n \\ \implies y_{n+1} &= \frac{1+h\left(1-2t_n\right)/2}{1-h\left(1-2t_{n+1}\right)/2}y_n \end{aligned}$$

Comparing the error of Trapezoidal method to that of Euler and Backward Euler, we have:

h	Euler	Backward Euler	Trapezoidal
0.25	0.23047	0.19036	0.0090254
0.125	0.10967	0.10177	0.0022883
0.0625	0.05405	0.051833	0.00057406
0.03125	0.02674	0.026218	0.00014364
0.015625	0.013308	0.013174	0.000035917

It is clear that the Trapezoidal rule much more accurate – moreover, notice that taking half a timestep smaller results in about a quarter of the error, as we would expect for a second order method.

4.2 Heun's Method

In some cases, it may not be possible to easily solve a step of the Trapezoidal method (12), particularly if $f(t,y)$ is nonlinear in y. To overcome this, we can instead consider a *two-stage* approach where we replace the implicit use of y_{n+1} with one step of Euler:

$$\begin{aligned} &\textit{Stage 1:} \quad y^*_{n+1} = y_n + hf\left(t_n, y_n\right) \\ &\textit{Stage 2:} \quad y_{n+1} = y_n + \frac{h}{2}\left(f\left(t_n,y_n\right) + f\left(t_{n+1}, y^*_{n+1}\right)\right) \end{aligned} \tag{13}$$

This is *Heun's method*. It is also a second order method, but note that the stability properties (not discussed in these notes) will have changed from that of Trapezoidal Rule. Notice we have achieved greater accuracy at the expense of more computational work: evaluating the function $f(t,y)$ twice, which could potentially be quite costly.

Example 4.2.1. Let us return once again to the IVP we visited in Example 2.1.1. Applying Heun's method is fairly straightforward, so let us jump to comparing the error of Heun's method to that of Euler, Backward Euler, and Trapezoidal Method:

h	Euler	Backward Euler	Trapezoidal	Heun
0.25	0.23047	0.19036	0.0090254	0.020025
0.125	0.10967	0.10177	0.0022883	0.0041702
0.0625	0.05405	0.051833	0.00057406	0.0009556
0.03125	0.02674	0.026218	0.00014364	0.00023048
0.015625	0.013308	0.013175	0.000035917	0.000056629

Notice how Heun's method is slightly less accurate than the Trapezoidal Method, but still follows the $O(h^2)$ behavior we expect.

Remark 4.2.1. If $f(t, y) = f(t)$, then there is no difference between Heun's method (13) and the Trapezoidal Method (12). Both are, in that case, just implementing the trapezoidal rule for numerical integration.

4.3 Runge-Kutta Methods

Euler's method (3) and Heun's Method (13) are part of a larger class of methods called *Runge-Kutta* methods. These methods evaluate $f(t, y)$ at many points along the interval $[t_n, t_{n+1}]$ to achieve greater accuracy, and are generally of the form:

$$y_{n+1} = y_n + h\, F\left(t_n, y_n; h\right) \tag{14}$$

where F defines multiple stage evaluations of $f(t, y)$. For example, the original "Runge-Kutta Method" is the 4-stage, 4th order RK method, which utilizes a weighted average of $f(t, y)$ values along the interval $t \in [t_n, t_{n+1}]$:

$$\begin{aligned} k_1 &= f\left(t_n, y_n\right) \\ k_2 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right) \\ k_3 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right) \\ k_4 &= f\left(t_n + h, y_n + hk_3\right) \\ y_{n+1} &= y_n + h\frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \end{aligned} \tag{15}$$

One can show that if $f(t, y) = f(t)$, RK4 (15) reduces to *Simpson's rule* for approximating the integral in (2).

Example 4.3.1. We would expect that solving the IVP of Example 2.1.1 with RK4 will be significantly more accurate to methods we have considered thus far.

h	Euler	Backward Euler	Trapezoidal	Heun	RK4
0.25	0.23047	0.19036	0.0090254	0.020025	5.1357e-4
0.125	0.10967	0.10177	0.0022883	0.0041702	2.4685e-5
0.0625	0.05405	0.051833	0.00057406	0.0009556	1.3451e-6
0.03125	0.02674	0.026218	0.00014364	0.00023048	7.8404e-8
0.015625	0.013308	0.013174	0.000035917	0.000056629	4.7318e-9

Indeed, we can see that the most accurate method for the IVP of Example 2.1.1 is RK4, and the error decays roughly by 1/16 when the timestep is multiplied by 1/2, as we would expect of a fourth order method.

4.4 Multistep Methods – Adams-Bashforth and Adams-Moulton

Thus far, we have only discussed *one-step methods*, where the approximation $y_{n+1} \approx \phi\left(t_{n+1}\right)$ is defined only by data at t_n. However, it is natural to ask: if we know all of the past steps $y_0, y_1, ..., y_n$, can we utilize that information to gain a more accurate approximation to the solution at t_{n+1} ?

This is the basis of multistep methods. Here, we will outline the ideas behind two sub-categories of multistep methods: Adams-Bashforth Methods, and Adams-Moulton Methods.

4.4.1 Adams-Bashforth

Recall that we have been considering numerical methods as derived from the integral equation formulation (2). We can rewrite the relevant integral of (2) equivalently as

$$\int_{t_n}^{t_{n+1}} f\left(t, \phi(t)\right)\, dt = \int_{t_n}^{t_{n+1}} \phi'(t)\, dt$$

Now, instead let us approach approximating the integral by approximating $\phi'(t)$ by a polynomial $P_k(t)$ of degree k. The primary idea underlying *Adams Methods* is to define the polynomial $P_k(t)$ by the time-value pairs $(t_i, \phi\left(t_i\right))$ for $i = n, n-1, ..., n-k$ (in total $k+1$ many points).

Consider for example the case of $k = 1$. We want to approximate $\phi'(t)$ with a polynomial $P_1(t) = At + B$ which satisfies

$$\begin{aligned} P_1\left(t_n\right) &= \phi'\left(t_n\right) = f\left(t_n, \phi\left(t_n\right)\right) \\ P_1\left(t_{n-1}\right) &= \phi'\left(t_{n-1}\right) = f\left(t_{n-1}, \phi\left(t_{n-1}\right)\right) \end{aligned}$$

These two conditions uniquely define the constants A and B. Evaluating the integral, you have

$$\int_{t_n}^{t_{n+1}} \phi'(t)\, dt \approx \int_{t_n}^{t_{n+1}} P_1(t)\, dt = \frac{A}{2}\left(t_{n+1}^2 - t_n^2\right) + B\left(t_{n+1} - t_n\right)$$

Then, after replacing $\phi\left(t_n\right)$ with its approximation y_n, we arrive at the *2nd order Adams Bashforth* method

$$y_{n+1} = y_n + \frac{3}{2} h f\left(t_n, y_n\right) - \frac{1}{2} h f\left(t_{n-1}, y_{n-1}\right) \tag{16}$$

If we instead let $k = 0$, necessarily you recover the Euler method (3). More accurate Adams formulas can be obtained by using higher-degree polynomials with more data points from previous steps, but notice that in multistep methods you will need to calculate the first $y_1, y_2, ..., y_k$ with some other method. Usually, one will use a onestep method of comparable accuracy to get the starting values.

4.4.2 Adams-Moulton

Adams-Moulton methods are similarly constructed to Adams-Bashforth methods: the integral is approximated by approximating $\phi'(t)$ by a polynomial $P_k(t)$ of degree k. However, this time the polynomial $P_k(t)$ is defined by the time-value pairs $(t_i, \phi\left(t_i\right))$ for $i = n{+}1, n, n-1, ..., n-k+1$, once again in total $k+1$ many points except now utilizing the t_{n+1} timepoint. This means that Adams-Moulton methods are *implicit*. Unsurprisingly, the first-order Adams-Moulton method is just Backward Euler (4), and the second-order Adams-Moulton method is just the Trapezoidal Method (12).