Euler's Method, Taylor Series Method, Runge Kutta Methods, Multi-Step Methods and Stability.

REVIEW: We start with the differential equation

$$\begin{aligned} \frac{dy(t)}{dt} &= f\left(t, y(t)\right) \\ y(0) &= y_0 \end{aligned} \tag{1.1}$$

This equation can be nonlinear, or even a system of nonlinear equations (in which case y is a vector and f is a vector of n different functions).

Numerical Solution of an ODE: The idea behind numerical solutions of a *Differential Equation* is to replace differentiation by differencing. A computer cannot differentiate but it can easily do a difference. (Differentiation is a continuous process. Differencing is a discrete process.) Now we introduce the most important tool that will be used in this section. By the time you've mastered this section, you'll be able to do **Taylor Expansions** in your sleep. (*I am already doing Taylor expansions in your sleep, right?!*)

Taylor Series Expansion: You'll recall (?) from your calculus class that if a function $y(t)$ behaves nicely enough, then its Taylor series expansion converges:

$$y(t + \Delta t) = y(t) + \Delta t y'(t) + \frac{1}{2}\Delta t^2 y''(t) + \frac{1}{3!}\Delta t^3 y'''(t) + ...$$

The Taylor series with remainder term is

$$y(t + \Delta t) = y(t) + \Delta t y'(t) + \frac{1}{2}\Delta t^2 y''(t) + \frac{1}{3!}\Delta t^3 y'''(t) + ... + \frac{1}{n!}\Delta t^n y^{(n)}(\tau)$$

where τ is some value between t and $t + \Delta t$. You can truncate this for any value of n.

Euler's Method: If we truncate the Taylor series at the first term

$$y(t + \Delta t) = y(t) + \Delta t y'(t) + \frac{1}{2}\Delta t^2 y''(\tau),$$

we can rearrange this and solve for $y'(t)$

$$y'(t) = \frac{y(t + \Delta t) - y(t)}{\Delta t} + O(\Delta t).$$

Now we can attempt to solve (1.1) by replacing the derivative with a difference:

$$y((n + 1)\Delta t) \approx y(n\Delta t) + \Delta t f(n\Delta t, y(n\Delta t))$$

Start with $y(0)$ and step forward to solve for any time.

What's good about this? If the O term is something nice looking, this quantity decays with Δt, so if we take Δt smaller and smaller, this gets closer and closer to the real value. What can go wrong? The O term may be ugly. The errors can accumulate as I step forward in time. Also, even though this may be a good approximation for $y'(t)$ it may not converge to the right solution. To answer these questions, we look at this scheme in depth.

Terminology: From now on, we'll call y_n the numerical approximation to the solution $y(n\Delta t)$; $t_n = n\Delta t$. Euler's method can then be written

$$y_{n+1} = y_n + \Delta t f(t_n, y_n) \qquad n = 1, ..., N-1 \tag{1.2}$$

This method assumes that you can move from one location to the next using the slope given by the equation (1.1). We saw last time that when we do this, our errors will decay linearly with Δt. We will show this again today, but in two steps, so that we can generalize it. The proof should look very familiar!

Local Truncation Error: To be able to evaluate what we expect the order of a method to look like, we look at the

$$LTE(t) = \frac{y(t+\Delta t) - y(t)}{\Delta t} - f(t, y(t)),$$

i.e. it is the residue when the exact solution of the ODE (1.1) is plugged into the numerical scheme. If y_n is close to $y(t_n)$ then the LTE will be close to zero.

The local truncation error represents the terms neglected by truncating the Taylor series. This is not the error that we get from the method, (i.e. the difference between the real solution and the numerical solution) but will be connected.

If I don't know $y(t)$, what is the use of this definition? (and if I do know $y(t)$, what do I need the method for?!). It turns out that even without explicit knowledge of the solution we can still calculate the LTE and use it as an estimate and control of the error, by placing certain smoothness assumptions on $y(t)$ and using the Taylor Expansions.

Clearly, at time t_n, Euler's method has **Local Truncation Error:**

$$LTE = \frac{y(t_n+\Delta t) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) = O(\Delta t),$$

in other words, we can write this

$$y(t_{n+1}) = y(t_n) + \Delta t f(t_n, y(t_n)) + \Delta t LTE.$$

Of course, the method is

$$y_{n+1} = y(t_n) + \Delta t f(t_n, y_n).$$

Subtract these two,

$$\begin{aligned}
|y(t_{n+1}) - y_{n+1}| &= |y(t_n) - y_n + \Delta t \left(f(t_n, y(t_n)) - f(t_n, y_n)\right) + \Delta t LTE| \\
&\leq |y(t_n) - y_n| + \Delta t \left|f(t_n, y(t_n)) - f(t_n, y_n)\right| + \Delta t |LTE| \\
&\leq |y(t_n) - y_n| + \Delta t L \left|y(t_n) - y_n\right| + \Delta t |LTE| \,.
\end{aligned}$$

Because f is Lipschitz continuous,

$$\left|\frac{f(t_n, y(t_n)) - f(t_n, y_n)}{y(t_n) - y_n}\right| \leq L.$$

And so, if we let the **Global Error** be $e_n = |y(t_n) - y_n|$, then we can bound the growth of this error:

$$e_{n+1} \leq e_n(1 + \Delta t L) + LTE\Delta t.$$

How does this help us bound the method?
Lemma: If $z_{i+1} \leq z_i(1 + a\Delta t) + b$ Then $z_i \leq e^{ai\Delta t}(z_0 + \frac{b}{a\Delta t})$
Proof:

$$\begin{aligned}
z_{i+1} \leq z_i \quad & (1 + a\Delta t) + b \\
\leq \quad & (z_{i-1}(1 + a\Delta t) + b)(1 + a\Delta t) + b \\
. \quad & \\
. \quad & \\
. \quad & \\
\leq \quad & z_0(1 + a\Delta t)^{i+1} + b(1 + (1 + a\Delta t)... + (1 + a\Delta t)^i) \\
= \quad & z_0(1 + a\Delta t)^{i+1} + b\frac{(1 + a\Delta t)^{i+1} - 1}{1 + a\Delta t - 1} \\
\leq \quad & z_0(1 + a\Delta t)^{i+1} + \frac{b}{a\Delta t}(1 + a\Delta t)^{i+1} \\
\leq \quad & (1 + a\Delta t)^{i+1}\left(z_0 + \frac{b}{a\Delta t}\right) \\
\leq \quad & e^{a\Delta t(i+1)}\left(z_0 + \frac{b}{a\Delta t}\right)
\end{aligned}$$

so

$$z_i \leq e^{ai\Delta t}(z_0 + \frac{b}{a\Delta t})$$

Applying this lemma to the global error, we have $|e_n| \leq e^{Ln\Delta t}(|e_0| + \frac{M}{2L}\Delta t)$ Now, if $n\Delta t \leq T$ then $|e_n| \leq e^{LT}(|e_0| + \frac{M}{2L}\Delta t)$ and since $|e_0| = 0$ we have:

$$|e_n| \leq e^{LT}(\frac{M}{2L}\Delta t).$$

Compare this with the local error: $LTE \leq \frac{1}{2}M\Delta t$ we see that the global error has the same order as the local error with a different coefficient in the estimates. They are related by the Lipschitz constant L and the final time T.

The **Order** of a scheme r, is defined by $|e_n| = O(\Delta t^r)$. The higher the order of the scheme, the faster the error decays.

Comment: The important thing to understand is that the **Local Truncation Error** is not always an indicator of what the **global error** will do. Schemes that have the same order of LTE and global error are good schemes. We need to define what makes the method have the property that the global error will be of same order as the LTE.

Taylor Series Methods: To derive these methods we start with a Taylor Expansion:

$$y(t+\Delta t) \approx y(t) + \Delta t y'(t) + \frac{1}{2}\Delta t^2 y''(t) + ... + \frac{1}{r!}y^{(r)}(t)\Delta t^r.$$

Let's say we want to truncate this at the second derivative and base a method on that. The scheme is, then:

$$y_{n+1} = y_n + f_n \Delta t + \frac{f'_{t_n}}{2}\Delta t^2.$$

The Taylor series method can be written as

$$y_{n+1} = y_n + \Delta t F(t_n, y_n, \Delta t)$$

where $F = f + \frac{1}{2}\Delta t f'$. If we take the LTE for this scheme, we get (as expected)

$$LTE(t) = \frac{y(t_n+\Delta t) - y(t_n)}{\Delta t} - f(t_n, y(t_n)) - \frac{1}{2}\Delta t f'(t_n, y(t_n)) = O(\Delta t^2).$$

Of course, we designed this method to give us this order, so it shouldn't be a surprise!

So the LTE is reasonable, but what about the global error? Just as in the Euler Forward case, we can show that the global error is of the same order as the LTE. How do we do this? We have two facts,

$$y(t_{n+1}) = y(t_n) + \Delta t F(t_n, y(t_n), \Delta t),$$

and

$$y_{n+1} = y_n + \Delta t F(t_n, y_n, \Delta t)$$

where $F = f + \frac{1}{2}\Delta t f'$. Now we subtract these two

$$\begin{aligned} |y(t_{n+1}) - y_{n+1}| &= |y(t_n) - y_n + \Delta t \left(F(t_n, y(t_n)) - F(t_n, y_n)\right) + \Delta t LTE| \\ &\leq |y(t_n) - y_n| + \Delta t \left|F(t_n, y(t_n)) - F(t_n, y_n)\right| + \Delta t |LTE| \,. \end{aligned}$$

Now, if F is Lipschitz continuous, we can say

$$e_{n+1} \leq (1 + \Delta t L)e_n + \Delta t |LTE|.$$

Of course, this is the same proof as for Euler's method, except that now we are looking at F, not f, and the LTE is of higher order. We can do this no matter which Taylor series method we use, how many terms we go forward before we truncate.

Advantages and Disadvantages of the Taylor Series Method:

advantages	a) One step, explicit
	b) can be high order
	c) easy to show that global error is the same order as LTE
disadvantages	Needs the explicit form of derivatives of f.

Runge-Kutta Methods To avoid the disadvantage of the Taylor series method, we can use Runge-Kutta methods. These are still **one step** methods, but they depend on estimates of the solution at different points. They are written out so that they don't look messy:

Second Order Runge-Kutta Methods:

$$\begin{aligned}
k_1 &= \Delta t f(t_i, y_i) \\
k_2 &= \Delta t f(t_i + \alpha\Delta t, y_i + \beta k_1) \\
y_{i+1} &= y_i + a k_1 + b k_2
\end{aligned}$$

let's see how we can chose the parameters a,b, α, β so that this method has the highest order LTE possible. Take the Taylor expansions to express the LTE:

$$\begin{aligned}
k_1(t) &= \Delta t f(t, y(t)) \\
k_2(t) &= \Delta t f(t + \alpha\Delta t, y + \beta k_1(t) \\
&= \Delta t \left(f(t, y(t) + f_t(t, y(t))\alpha\Delta t + f_y(t, y(t))\beta k_1(t) + O(\Delta t^2) \right) \\
LTE(t) &= \frac{y(t+\Delta t) - y(t)}{\Delta t} - \frac{a}{\Delta t} f(t, y(t))\Delta t - \frac{b}{\Delta t}\left(f_t(t, y(t))\alpha\Delta t + f_y(t, y(t)\beta k_1(t)\right. \\
&+ \left. f(t, y(t))\,\Delta t + O(\Delta t^2)\right) \\
&= \frac{y(t+\Delta t) - y(t)}{\Delta t} - a f(t, y(t)) - b f(t, y(t)) - b f_t(t, y(t))\alpha \\
&- b f_y(t, y(t)\beta f(t, y(t)) + O(\Delta t^2) \\
&= y'(t) + \frac{1}{2}\Delta t y''(t) - (a+b) f(t, y(t)) - \Delta t(b\alpha f_t(t, y(t)) + b\beta f(t, y(t)) f_y(t, y(t)) + O(\Delta t^2) \\
&= (1 - a - b) f + (\frac{1}{2} - b\alpha)\Delta t f_t + (\frac{1}{2} - b\beta)\Delta t f_y f + O(\Delta t^2)
\end{aligned}$$

So we want $a = 1 - b$, $\alpha = \beta = \frac{1}{2b}$.

Fourth Order Runge-Kutta Methods:

$$k_1 = \Delta t f(t_i, y_i) \tag{1.3}$$
$$k_2 = \Delta t f(t_i + \frac{1}{2}\Delta t, y_i + \frac{1}{2}k_1) \tag{1.4}$$
$$k_3 = \Delta t f(t_i + \frac{1}{2}\Delta t, y_i + \frac{1}{2}k_2) \tag{1.5}$$
$$k_4 = \Delta t f(t_i + \Delta t, y_i + k_3) \tag{1.6}$$
$$y_{i+1} = y_i + \frac{1}{6}(k_1 + k_2 + k_3 + k_4) \tag{1.7}$$

The second order method requires 2 evaluations of f at every timestep, the fourth order method requires 4 evaluations of f at every timestep. In general: **For an rth order Runge-Kutta method we need S(r) evaluations of f for each timestep, where**

$$S(r) = \begin{cases} r & \text{for } r \leq 4 \\ r+1 & \text{for } r = 5 \text{ and } r = 6 \\ \geq r+2 & \text{for } r \geq 7 \end{cases}$$

Practically speaking, people stop at $r = 5$.

Advantages of Runge-Kutta Methods

1. One step method – global error is of the same order as local error.
2. Don't need to know derivatives of f.
3. Easy for "Automatic Error Control".

Automatic Error Control Uniform grid spacing – in this case, time steps – are good for some cases but not always. Sometimes we deal with problems where varying the gridsize makes sense. How do you know when to change the stepsize? If we have an rth order scheme and and $r + 1$th order scheme, we can take the difference between these two to be the error in the scheme, and make the stepsize smaller if we prefer a smaller error, or larger if we can tolerate a larger error.

For Automatic error control yo are computing a "useless" (r+1)th order shceme . . . what a waste! But with Runge Kutta we can take a fifth order method and a fourth order method, using the same ks. only a little extra work at each step.