THEORY AND METHODS FOR ONE-STEP ODES

David F. Gleich

April 20, 2021

These notes are based on sections 5.3, 5.4, 5.5, 5.6, and 5.7 in Gautschi's Numerical Analysis textbook..

THE PROBLEM

We are considering numerical methods for the initial value problem:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}) \qquad \mathbf{y}(0) = \mathbf{y}_0, t \in [0, T]. \tag{1}$$

where $\mathbf{f}$ is continuous and outputs an $\mathbb{R}^d$ vector.

EXISTENCE AND UNIQUENESS

THEOREM 1 (Gaustchi, p.331, Theorem 5.3.1) [1] *Assume that $\mathbf{f}(t, \mathbf{y})$ is continuous in the first variable (t) in the range $[0, T]$ and with respect to the second variable ($\mathbf{y}$), we satisfy a uniform Lipschitz condition:*

$$\|\mathbf{f}(t, \mathbf{y}_1) - \mathbf{f}(t, \mathbf{y}_2)\| \le L\|\mathbf{y}_1 - \mathbf{y}_2\|, \qquad t \in [0, T], \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{R}^d.$$

(The norm can be arbitrary) Then the problem (1) has a unique solution $\mathbf{y}(t), 0 \le t \le T$ for arbitrary $\mathbf{y}_0$ and the solution depends continuously on $\mathbf{y}_0$.

[1] This theorem is called the Picard-Lindelöf theorem,https://en.wikipedia.org/wiki/Picard%E2%80%93Lindel%C3%B6f_theorem

The only hard part about this statement is the Lipschitz condition. This is called Lipschitz continuity too.[2]

[2] See Wikipedia https://en.wikipedia.org/wiki/Lipschitz_continuity.

Intuitive Figure of Lipschitz

It's actually really hard to satisfy. The functions $f(x) = x^2$ and $f(x) = e^x$ do not satisfy this requirement for all $\mathbb{R}$. But $\sin(x)$ and $\cos(x)$ do. The reason these functions are okay is that they are nicely behaved for all input x, whereas x^2 and e^x are "infinitely" steep as $x \to \infty$.

A slightly weaker condition is *locally Lipschitz*, which would suffice for the uniqueness part, but not existence. The reason is that it's possible for $\mathbf{y}(t) \to \pm\infty$ in finite time. This would prohibit having a solution for an arbitrary time T.[3]

[3] http://math.stackexchange.com/questions/1441492/is-local-lipschitz-continuity-sufficient-for-an-ode-to-have-a-unique-solution

EXAMPLE 2 *That page has a great example, which is*

$$dy/dt = y^2, \quad y(0) = 1, \qquad then \qquad y(t) = 1/(1-t).$$

This function does not exist at $t = 1$.

Also, it turns out that continuity is not required for existence. This is handled by the Carathéodory theorem.[4]

But suffice it to say, for this class, we can assume things are pretty nice!

[4] https://en.wikipedia.org/wiki/Carath%C3%A9odory%27s_existence_theorem

GRID APPROXIMATIONS

The methods we will consider in this class are all grid-approximations of the function $\mathbf{y}(t)$. That is, we consider

$$\mathbf{y}(t) \approx \mathbf{y}(0), \mathbf{y}(t_1), \ldots, \mathbf{y}(t_N) \qquad t_N = T$$

and usually uniformly spaced grids where $t_i = ih$ for some $h = T/N$.

Let $\mathbf{y}_i = \mathbf{y}(t_i)$ for convenience.

Notation
N is the number of "time-steps"
$h = T/N$ is the grid-size
$\mathbf{y}_i = \mathbf{y}(t_i)$ is shorthand.

SPECTRAL APPROXIMATIONS

We'll see spectral approximations, where we represent $y_i(t)$ as a polynomial, soon!

ONE-STEP METHODS

We have seen two methods already.

Method	(Alternate Name)	Update equation
Forward Euler	Explicit Euler	$\mathbf{y}_{i+1} = \mathbf{y}_i + h\mathbf{f}(ih, \mathbf{y}_i)$
Backward Euler	Implicit Euler	Solve $\mathbf{y}_{i+1} = \mathbf{y}_i + h\mathbf{f}((i+1)h, \mathbf{y}_{i+1})$

My names for these are explicit first-order extrapolation (instead of forward Euler) or forward difference extrapolation (instead of forward Euler) and implicit backward differencing (instead of backward Euler). These are both one-step methods that relate $\mathbf{y}_{i+1}$ to $\mathbf{y}_i$.

For the moment, we'll only consider *explicit* methods, those that do not depend on solving systems of equations such as the Backward or Implicit methods.

In general, a one-step method is[5]

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h\mathbf{\Phi}(ih, \mathbf{y}_i; h).$$

Which we'll also write as:

$$\mathbf{y}_+ = \mathbf{y} + h\mathbf{\Phi}(t, \mathbf{y}; h)$$

to avoid the index i and make it slightly more general.

For explicit Euler, $\mathbf{\Phi} = \mathbf{f}(t, \mathbf{y})$.

The idea with $\mathbf{\Phi}$ is that we should locally approximate the initial value problem

$$\frac{d\mathbf{u}}{d\tau} = \mathbf{f}(\tau, \mathbf{u}), \qquad t \le \tau \le t + h, \quad \mathbf{u}(t) = \mathbf{y}$$

[5] The notation $\mathbf{\Phi}(t, \mathbf{y}; h)$ just means that $\mathbf{\Phi}$ is a function that *knows* the value of h, but can do essentially anything with that information. My take, it is not a function of h in the mathematical sense, but it is a function of h in the computer science sense or algorithm sense. This notation is often used to describe "parameters" that are some how "outside" of the approximation problem itself.

Intuitive Figure of One-step methods

ERROR ANALYSIS: LOCAL TRUNCATION

Now it's time to talk error! Here, we'll use $\mathbf{u}(\tau)$ as the reference solution. We want to get as close as possible to this!

DEFINITION 3 *The truncation error of the method* $\mathbf{\Phi}$ *with respect to* $\mathbf{u}$ *is:*

$$\boldsymbol{T}(t,\mathbf{y};h) = \frac{1}{h}(\mathbf{y}_{+} - \mathbf{u}(t+h)).$$

This is how much we are different than the true solution. We can use our function $\mathbf{\Phi}$ to write:

$$\boldsymbol{T}(t,\mathbf{y};h) = \mathbf{\Phi}(t,\mathbf{y};h) - \frac{1}{h}[\mathbf{u}(t+h) - \mathbf{u}(t)].$$

- A method is consistent if $\boldsymbol{T}(t,\mathbf{y};h) \to 0$ as $h \to 0$ for all $t, \mathbf{y}$ in some domain, uniformly.
- Consistency is equivalent to $\mathbf{\Phi}(t,\mathbf{y};0) = \mathbf{f}(t,\mathbf{y})$.
- A method has *order* p if (for some vector norm) $\|\boldsymbol{T}(t,\mathbf{y};h)\| \leq Ch^p$ for some constant.

The final concept we'll use is the *principal error function*. This is a function θ such that:

$$T(t,\mathbf{y};h) = \theta(t,\mathbf{y})h^p + O(h^{p+1}) \qquad h \to 0.$$

so that θ is the leading term of the error function and we require $\theta \neq 0$ at all points.

Notes This analysis is all local, just in the region around *one point* in space and time. We'll see global analysis later.

METHODS

FORWARD EULER

We already saw this, let's study its error.

Since $\mathbf{\Phi}(t,\mathbf{y};h) = \mathbf{f}(t,\mathbf{y})$, it's clear that the method is consistent.

The method has order 1.

TAYLOR EXTRAPOLATION

Recall that we could derive forward Euler by approximating:

$$\frac{d\mathbf{y}}{dt} \approx 1/h(\mathbf{y}_{i+1} - \mathbf{y}_i).$$

We could also have used a different type of approximation based on the Taylor series:

$$\mathbf{y}(t+h) \approx \mathbf{y}(t) + h\mathbf{y}'(t) + O(h^2)$$

and then solving for $\mathbf{y}(t+h)$ given that $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$. This suggests that we could use higher-order Taylor expansion:

$$\mathbf{y}(t+h) \approx \mathbf{y}(t) + h\mathbf{y}'(t) + h^2/2\mathbf{y}''(t) + h^3/6\mathbf{y}'''(t) + O(h^4).$$

Intuitive Figure of Taylor approximation

In this method, we use additional terms from the Taylor series and use:

$$\boldsymbol{\Phi}(t, \mathbf{y}; h) = \mathbf{f}(t, \mathbf{y}) + \frac{1}{2}h\mathbf{f}'(t, \mathbf{y}) + \frac{1}{3!}h^2\mathbf{f}''(t, \mathbf{y}) + \ldots + \frac{1}{p!}h^{p-1}\mathbf{f}^{[p-1]}(t; \mathbf{y}).$$

This method is order p.

This method require additional derivatives of the function $\mathbf{f}$, which may be hard get.

IMPROVING EULER WITH TWO-STAGE METHODS.

Intuitive Figure of Mid-point Euler and Heun's method

The idea here is that explicit Euler is too aggressive. We need something better!

So we follow the derivative given by Euler for $h/2$ time, then update our estimate of the derivative, and "go-back" and follow that over the entire time-span.

1. Go forward in time $h/2$: $\mathbf{y}(t+h/2) \approx \mathbf{y}(t) + h/2\mathbf{f}(t, \mathbf{y}(t))$
2. Get the derivative at $t + h/2$: $\mathbf{p} \approx \mathbf{f}(t + h/2, \mathbf{y}(t + h/2))$

3. Then follow $\mathbf{p}$ over the entire span: $\mathbf{y}_+ = \mathbf{y} + h\mathbf{p}$.

We can wrap this into:

$$\mathbf{y}_+ = \mathbf{y} + h\mathbf{f}(t + h/2, \mathbf{y} + h/2\mathbf{f}(t, \mathbf{y})),$$

which is a one-step method with

$$\mathbf{\Phi}(t, \mathbf{y}; h) = \mathbf{f}(t + h/2, \mathbf{y} + h/2\mathbf{f}(t, \mathbf{y})).$$

As $h \to 0$, again, we get consistency.

In this case, we could also have gone to $t+h$, and taken the average of the slopes which gives Heun's method.

This suggests a general scheme.

1. Compute $\mathbf{k}_1$, the slope at $\mathbf{y}$, t
2. Compute $\mathbf{k}_2$, the slope at some point $t + \mu h$ in between $t, t + h$ based on $\mathbf{k}_1$
3. Then use $\mathbf{\Phi} = \alpha_1 \mathbf{k}_1 + \alpha \mathbf{k}_2$.

This gives 3 parameters; α_1, α_2, μ. We can optimize over these parameters to seek the higher-order method!

The book does all the analysis here in grueling detail. It involves a number of steps of Taylor's analysis. But the important point is that we get an order 2 method if:

$$\alpha_1 + \alpha_2 = 1 \text{ and } \alpha_2 \mu = 1/2.$$

So this handles the mid-point Euler method and Heun's method nicely.

RUNGE-KUTTA SCHEMES

The most general scheme are the RK (Runge-Kutta) integrators. These take the previous idea to a general setting.

$$\begin{aligned}
\mathbf{\Phi}(t, \mathbf{y}; h) &= \sum_{s=1}^{r} \alpha_s \mathbf{k}_s \\
\mathbf{k}_1 &= \mathbf{f}(t, \mathbf{y}) \\
\mathbf{k}_s &= \mathbf{f}(t + \mu_s, \mathbf{y} + h \sum_{j=1}^{s-1} \lambda_{sj} \mathbf{k}_j).
\end{aligned}$$

Here, we have $\mu_s = \sum_{j=1}^{s-1} \lambda_{sj}$ and $\sum \alpha_s = 1$.

Through some extensive analysis, the method has error of order r for $1 \le r \le 4$. If $r = 8$ or $r = 9$, then we can get order 6 and order 7 methods.

For instance, the *classic* formula is order 4 with:

$$\begin{aligned}
\mathbf{\Phi}(t, \mathbf{y}; h) &= \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \\
\mathbf{k}_1 &= \mathbf{f}(t, \mathbf{y}) \\
\mathbf{k}_2 &= \mathbf{f}(t + h/2, \mathbf{y} + h/2\mathbf{k}_1) \\
\mathbf{k}_3 &= \mathbf{f}(t + h/2, \mathbf{y} + h/2\mathbf{k}_2) \\
\mathbf{k}_4 &= \mathbf{f}(t + h, \mathbf{y} + h\mathbf{k}_3)
\end{aligned}$$

SOFTWARE

Matlab's ODE suite implements:[6]

1. `ode45`: 4th-order method (best general choice)

[6] For more see, `http://blogs.mathworks.com/cleve/2014/05/26/ordinary-differential-equation-solvers-ode23-and-ode45/`

2. `ode23`: 2nd-order method

3. `ode23s`: 2nd-order method tuned for stiff problems using a Jacobian.

4. `ode15s`: 1st-order method tuned for stiff problem

This was designed by Shampine and Bogacki.

Julia has a suite of ODE solvers that's based on the same ideas as Matlab in the `ODE` package.[7] The methods are the same as Matlab.

SciPy has many of the same methods implemented.[8]

[7] https://github.com/JuliaLang/ODE.jl

[8] http://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html