THEORY AND METHODS FOR ONE-STEP ODES

David F. Gleich

April 22, 2021

These notes are based on sections 5.7, 5.8, and 5.9 in Gautschi's Numerical Analysis textbook..

OUTLINE

In this set of notes, we'll work towards getting estimates of the error involved in approximating the solution of an ODE and also how we can use this to guide step size selection dynamically. Finally, we'll mention what a stiff problem is.

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.

APPROXIMATION ON GRID FUNCTIONS

Let $\mathbf{y}(t)$ be the true solution. Let $\mathbb{H}$ be the grid defined by step-sizes: $h_1, \ldots, h_N$ such that

$$0 < t_1 < \ldots < t_N$$

satisfies $t_i = \sum_{\ell=1}^{i} h_i$.[1] Let $|\mathbb{H}| = \max_i h_i$.

Let $\mathbf{y}_{\mathbb{H}}$ be the value of $\mathbf{y}(t)$ at each of the points in the grid $\mathbb{H}$. We think of this as a set of vectors $\mathbf{y}_k = \mathbf{y}(t_k)$, where each $\mathbf{y}_k \in \mathbb{R}^d$.

Let $\Gamma_{\mathbb{H}}$ be the space of grid-functions.[2] Let $\mathbf{z} \in \Gamma_{\mathbb{H}}$, then[3]

$$\|\mathbf{z}\|_\infty = \max_i \|\mathbf{z}_i\|.$$

[1] This is just the obvious grid where $h_i = t_i - t_{i-1}$. We are parameterizing it by the step-sizes h_i instead of times because h is what shows up in the theory.

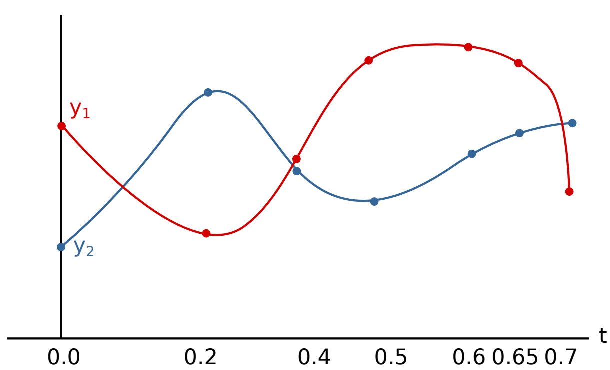
[2] You can think of this as the space of $d \times N$ matrices where each column is a time-step.

[3] The choice of norm inside the max is arbitrary

We can look at the difference between two grid functions via this norm:

$$\|\mathbf{z} - \mathbf{w}\|_\infty \qquad \mathbf{z}, \mathbf{w} \in \Gamma_{\mathbb{H}}$$

This grid is missing the zero point.



EXAMPLE 1

$$\mathbb{H} = (0, 0.2, 0.4, 0.5, 0.6, 0.65, 0.7)$$

$$\mathbf{y}_{\mathbb{H}} = \left(\begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0.6 \\ 1.75 \end{bmatrix}, \begin{bmatrix} 1.1 \\ 0.9 \end{bmatrix}, \begin{bmatrix} 1.85 \\ 0.75 \end{bmatrix}, \begin{bmatrix} 1.9 \\ 1.1 \end{bmatrix}, \begin{bmatrix} 1.8 \\ 1.25 \end{bmatrix}, \begin{bmatrix} 0.85 \\ 1.35 \end{bmatrix} \right)$$

Notes on notation The choice of grid is "given" by the space $\Gamma_{\mathbb{H}}$, so we can't compare functions drawn from different grids.

CONVERGENCE OF GLOBAL ERROR

In this set of notes, our goal will be to understand the following notion of error convergence.

DEFINITION 2 (5.7.2) *Let $\mathbf{u}$ be the result of approximating $\mathbf{y}(t)$ via a method $\mathbf{\Phi}$ on a grid $\mathbb{H}$. Then we say that $\mathbf{\Phi}$ converges if*

$$\|\mathbf{u} - \mathbf{y}_{\mathbb{H}}\|_\infty \to 0 \qquad \textit{as } |\mathbb{H}| \to 0.$$

THE KEY POINT

There are two properties that will let us show convergence:

- Local consistency
- Global stability.

If a method $\mathbf{\Phi}$ has both of these, then it will converge.

We saw local consistency in the last class, which just involves showing that as $h \to 0$, $\mathbf{\Phi}(t, \mathbf{y}; h) = \mathbf{f}(t, \mathbf{y})$.

STABILITY

The idea with stability is that we want to make sure that if the method $\mathbf{\Phi}$ sees a small change, then it only makes a small change in the result.

Let $\mathbb{H}$ be an arbitrary grid and $\mathbf{u}$ be a function on that grid. The system of equations that an ODE with step $\mathbf{\Phi}$ satisfies is

$$(\boldsymbol{R}_{\mathbb{H}}\mathbf{u}) = 0$$

where $(\boldsymbol{R}_{\mathbb{H}}\mathbf{u})_i = \frac{1}{h_i}(\mathbf{u}_{i+1} - \mathbf{u}_i) - \mathbf{\Phi}(t_i, \mathbf{u}_i; h_i)$ is a block of variables. [4]

[4] Show an example here of what we mean... This is just an example of that big block of equations we solved for Forward Euler and Backward Euler on our test problem.

Example of what is happening

Consider a grid function $\mathbf{u}$ and $\mathbf{w}$ that result from using a method $\mathbf{\Phi}$ with $\mathbf{u}_0$ and $\mathbf{w}_0$ as the initial conditions. Then, a method is stable if

$$\|\mathbf{u} - \mathbf{w}\|_\infty \le K(\|\mathbf{u}_0 - \mathbf{w}_0\| + \|\boldsymbol{R}_{\mathbb{H}}\mathbf{u} - \boldsymbol{R}_{\mathbb{H}}\mathbf{w}\|_\infty)$$

This definition isn't much fun to work with. So we show a sufficient condition

THEOREM 3 (5.7.1) *If $\mathbf{\Phi}(t, \mathbf{y}; h)$ satisfies the Lipschitz condition*

$$\|\mathbf{\Phi}(t, \mathbf{z}; h) - \mathbf{\Phi}(t, \mathbf{w}; h)\| \le M\|\mathbf{z} - \mathbf{w}\|$$

for all $0 \le t \le T$ and $0 < h < h^*$ and for all $\mathbf{z}, \mathbf{w}$. Then the method $\mathbf{\Phi}$ is stable.

EXAMPLE 4 *Consider the Forward Euler method applied to $dy/dt = y^2$ with $y(0) = 1$.. Then, if we use Forward Euler, we will get a finite value at $y(1)$. But the true value is infinite, so we do not satisfy a global error condition. (Of course, the singularity will be evident in the plot, but this won't confirm that it exists in the true solution, but will give a strong hint.)*

See Julia example!

CONVERGENCE

THEOREM 5 (5.7.2) *If a method $\mathbf{\Phi}$ is consistent and stable, then it converges. Moreover, if the method $\mathbf{\Phi}$ has order p, then the global error*

$$\|\mathbf{u} - \mathbf{y}_{\mathbb{H}}\|_{\infty} = O(|\mathbb{H}|^p).$$

STEP SIZE CONTROL

While we have introduced the methods using a uniform step size, in practice, it is usually helpful to have a variable step-size. One way of acccomplishing this is to estimate the local truncation error function directly. Then we'll adjust the local step-size to ensure that the local truncation error stays small. This will ensure we can use large steps when possible and small steps when necessary. In the book, we'll discuss embedded methods.

The essential idea is to computationally (and coursely) estimate the principal error function that results from locally using a time-step h.

Recall 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 $\theta \not\equiv 0$ at all points.

Consider two methods $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ such that $\mathbf{\Phi}_1$ has error local truncation p and $\mathbf{\Phi}_2$ has local truncation error $p + 1$ and look at the difference:

$$\mathbf{r}(t, \mathbf{y}; h) = \frac{1}{h^p}(\mathbf{\Phi}_1(t, \mathbf{y}; h) - \mathbf{\Phi}_2(t, \mathbf{y}; h)).$$

This suffices because:

$$\mathbf{\Phi}_1(t, \mathbf{y}; h) - \frac{1}{h}[\mathbf{u}(t + h) - \mathbf{u}(t)] = \theta(t, \mathbf{y})h^p + O(h^{p+1})$$
$$\mathbf{\Phi}_2(t, \mathbf{y}; h) - \frac{1}{h}[\mathbf{u}(t + h) - \mathbf{u}(t)] = O(h^{p+1})$$

The key idea in making this work is to find pairs of methods that share many function evaluations $\mathbf{f}$. In the 60s, a number of individuals worked out pairs of Runge-Kutta functions that do exactly this. Hence, `ode45` uses a 5th order method to get the error estimate on a 4th order method.

If we pick h_i such that the local trunction error is always a constant, then we do get a precise (but hard to compute) bound on the global error, that satisfy a linear relationship.

So here's a strategy that works

1. At time step i
2. Estimate h_i (use a larger value is the previous check passed on the first time), otherwise, use the last value.
3. Compute $\mathbf{u}_{i+1}$ using h_i and $\mathbf{\Phi}$.
4. Check $\|\mathbf{r}\| = \|\mathbf{\Phi}_1 - \mathbf{\Phi}_2\|$ and if $\|\mathbf{r}\| \geq \varepsilon$ then decrease h_i, and keep doing so until the test passes.[5]

[5] So we are saying that if we don't change the answer much by using a higher-order method, then we have sufficient accuracy at this point.

STIFF PROBLEMS

Some ODE problems are called *stiff* what this means is that we have multiple features. A common scenario is some time of slowly changing function but also something that oscillates really quickly. A good example would be the global climate. We have daily fluctuations in global temperature (at each point) and then also long-term climate shifts.

But these features are even present in simple linear ODEs. (See the Julia example)