AM 205: lecture 12

- Last time: Numerical differentiation, numerical solution of ordinary differential equations
- Today: Convergence and stability of ODE numerical methods
- Reminder: assignment 2 due tomorrow at 5 PM

Convergence

We define global error, e_k, as the total accumulated error at $t = t_k$

$$e_k \equiv y(t_k) - y_k$$

We define truncation error, T_k, as the amount "left over" at step k when we apply our method to the exact solution and divide by h

e.g. for an explicit one-step ODE approximation, we have

$$T_k \equiv \frac{y(t_{k+1}) - y(t_k)}{h} - \Phi(t_k, y(t_k); h)$$

Convergence

The truncation error defined above determines the local error introduced by the ODE approximation

For example, suppose $y_k = y(t_k)$, then for the case above we have

$$hT_k \equiv y(t_{k+1}) - y_k - h\Phi(t_k, y_k; h) = y(t_{k+1}) - y_{k+1}$$

Hence hT_k is the error introduced in one step of our ODE approximation[1]

Therefore the global error e_k is determined by the accumulation of the T_j for $j = 0, 1, \ldots, k-1$

Now let's consider the global error of the Euler method in detail

[1] Because of this fact, the truncation error is defined as hT_k in some texts

Convergence

Theorem: Suppose we apply Euler's method for steps $1, 2, \ldots, M$, to $y' = f(t, y)$, where f satisfies a Lipschitz condition:

$$|f(t, u) - f(t, v)| \leq L_f |u - v|,$$

where $L_f \in \mathbb{R}_{>0}$ is called a Lipschitz constant. Then

$$|e_k| \leq \frac{\left(e^{L_f t_k} - 1\right)}{L_f} \left[\max_{0 \leq j \leq k-1} |T_j| \right], k = 0, 1, \ldots, M,$$

where T_j is the Euler method truncation error.[2]

[2] Notation used here supposes that $y \in \mathbb{R}$, but the result generalizes naturally to $y \in \mathbb{R}^n$ for $n > 1$

Convergence

Proof: From the definition of truncation error for Euler's method we have

$$y(t_{k+1}) = y(t_k) + hf(t_k, y(t_k); h) + hT_k$$

Subtracting $y_{k+1} = y_k + hf(t_k, y_k; h)$ gives

$$e_{k+1} = e_k + h\left[f(t_k, y(t_k)) - f(t_k, y_k)\right] + hT_k,$$

hence

$$|e_{k+1}| \leq |e_k| + hL_f|e_k| + h|T_k| = (1 + hL_f)|e_k| + h|T_k|$$

Convergence

Proof (continued...):

This gives a geometric progression, *e.g.* for $k = 2$ we have

$$\begin{aligned} |e_3| &\leq (1 + hL_f)|e_2| + h|T_2| \\ &\leq (1 + hL_f)((1 + hL_f)|e_1| + h|T_1|) + h|T_2| \\ &\leq (1 + hL_f)^2 h|T_0| + (1 + hL_f)h|T_1| + h|T_2| \\ &\leq h\left[\max_{0\leq j\leq 2} |T_j|\right] \sum_{j=0}^{2} (1 + hL_f)^j \end{aligned}$$

Or, in general

$$|e_k| \leq h\left[\max_{0\leq j\leq k-1} |T_j|\right] \sum_{j=0}^{k-1} (1 + hL_f)^j$$

Convergence

Proof (continued...):

Hence use the formula

$$\sum_{j=0}^{k-1} r^j = \frac{1-r^k}{1-r}$$

with $r \equiv (1 + hL_f)$, to get

$$|e_k| \leq \frac{1}{L_f} \left[\max_{0 \leq j \leq k-1} |T_j| \right] ((1 + hL_f)^k - 1)$$

Finally, we use the bound[3] $1 + hL_f \leq \exp(hL_f)$ to get the desired result. $\square$

[3] For $x \geq 0$, $1 + x \leq \exp(x)$ by power series expansion $1 + x + x^2/2 + \cdots$

Convergence: Lipschitz Condition

A simple case where we can calculate a Lipschitz constant is if $y \in \mathbb{R}$ and f is continuously differentiable

Then from the mean value theorem we have:

$$|f(t,u) - f(t,v)| = |f_y(t,\theta)||u - v|,$$

for $\theta \in (u,v)$

Hence we can set:

$$L_f = \max_{\substack{t\in[0,t_M] \\ \theta\in(u,v)}} |f_y(t,\theta)|$$

Convergence: Lipschitz Condition

However, f doesn't have to be continuously differentiable to satisfy Lipschitz condition!

e.g. let $f(x) = |x|$, then $|f(x) - f(y)| = ||x| - |y|| \leq |x - y|$,[4] hence $L_f = 1$ in this case

[4] This is the reverse triangle inequality

Convergence

For a fixed t (*i.e.* $t = kh$, as $h \to 0$ and $k \to \infty$), the factor $(e^{L_f t} - 1)/L_f$ in the bound is a constant

Hence the global convergence rate for each fixed t is given by the dependence of T_k on h

Our proof was for Euler's method, but the same dependence of global error on local error holds in general

We say that a method has order of accuracy p if $|T_k| = O(h^p)$ (where p is an integer)

Hence ODE methods with order ≥ 1 are convergent

Order of Accuracy

Forward Euler is first order accurate:

$$
\begin{aligned}
T_k &\equiv \frac{y(t_{k+1}) - y(t_k)}{h} - f(t_k, y(t_k)) \\
&= \frac{y(t_{k+1}) - y(t_k)}{h} - y'(t_k) \\
&= \frac{y(t_k) + hy'(t_k) + h^2 y''(\theta)/2 - y(t_k)}{h} - y'(t_k) \\
&= \frac{h}{2} y''(\theta)
\end{aligned}
$$

Order of Accuracy

Backward Euler is first order accurate:

$$
\begin{aligned}
T_k &\equiv \frac{y(t_{k+1}) - y(t_k)}{h} - f(t_{k+1}, y(t_{k+1})) \\
&= \frac{y(t_{k+1}) - y(t_k)}{h} - y'(t_{k+1}) \\
&= \frac{y(t_{k+1}) - y(t_{k+1}) + hy'(t_{k+1}) - h^2 y''(\theta)/2}{h} - y'(t_{k+1}) \\
&= -\frac{h}{2} y''(\theta)
\end{aligned}
$$

Order of Accuracy

Trapezoid method is second order accurate:

Let's prove this using a quadrature error bound, recall that:

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(s, y(s))\mathrm{d}s$$

and hence

$$\frac{y(t_{k+1}) - y(t_k)}{h} = \frac{1}{h}\int_{t_k}^{t_{k+1}} f(s, y(s))\mathrm{d}s$$

So

$$T_k = \frac{1}{h}\int_{t_k}^{t_{k+1}} f(s, y(s))\mathrm{d}s - \frac{1}{2}\left[f(t_k, y(t_k)) + f(t_{k+1}, y(t_{k+1}))\right]$$

Order of Accuracy

Hence

$$\begin{aligned} T_k &= \frac{1}{h}\left[\int_{t_k}^{t_{k+1}} f(s, y(s))\mathrm{d}s - \frac{h}{2}\left(f(t_k, y(t_k)) + f(t_{k+1}, y(t_{k+1}))\right)\right] \\ &= \frac{1}{h}\left[\int_{t_k}^{t_{k+1}} y'(s)\mathrm{d}s - \frac{h}{2}\left(y'(t_k) + y'(t_{k+1})\right)\right] \end{aligned}$$

Therefore T_k is determined by the trapezoid rule error for the integrand y' on $t \in [t_k, t_{k+1}]$

Recall that trapezoid quadrature rule error bound depended on $(b-a)^3 = (t_{k+1} - t_k)^3 = h^3$ and hence

$$T_k = O(h^2)$$

Order of Accuracy

The table below shows global error at $t = 1$ for $y' = y$, $y(0) = 1$ for (forward) Euler and trapezoid

h	E_{Euler}	E_{Trap}
2.0e-2	2.67e-2	9.06e-05
1.0e-2	1.35e-2	2.26e-05
5.0e-3	6.76e-3	5.66e-06
2.5e-3	3.39e-3	1.41e-06

$$h \to h/2 \implies E_{\text{Euler}} \to E_{\text{Euler}}/2$$

$$h \to h/2 \implies E_{\text{Trap}} \to E_{\text{Trap}}/4$$

Stability

So far we have discussed convergence of numerical methods for ODE IVPs, *i.e.* asymptotic behavior as $h \to 0$

It is also crucial to consider stability of numerical methods: for what (finite and practical) values of h is our method stable?

We want our method to be well-behaved for as large a step size as possible

All else being equal, larger step sizes $\implies$ fewer time steps $\implies$ more efficient!

Stability

In this context, the key idea is that we want our methods to inherit the stability properties of the ODE

If an ODE is unstable, then we can't expect our discretization to be stable

But if an ODE is stable, we want our discretization to be stable as well

Hence we first discuss ODE stability, independent of numerical discretization

ODE Stability

Consider an ODE $y' = f(t, y)$, and

- Let $y(t)$ be the solution for initial condition $y(0) = y_0$
- Let $\hat{y}(t)$ be the solution for initial condition $\hat{y}(0) = \hat{y}_0$

The ODE is stable if:

For every $\epsilon > 0$, $\exists \delta > 0$ such that

$$\|\hat{y}_0 - y_0\| \leq \delta \implies \|\hat{y}(t) - y(t)\| \leq \epsilon$$

for all $t \geq 0$

"Small input perturbation leads to small perturbation in the solution"

ODE Stability

Stronger form of stability, asymptotic stability: $\|\hat{y}(t) - y(t)\| \to 0$ as $t \to \infty$, perturbations decay over time

These two definitions of stability are properties of the ODE, independent of any numerical algorithm

This nomenclature is a bit confusing compared to previous Units:

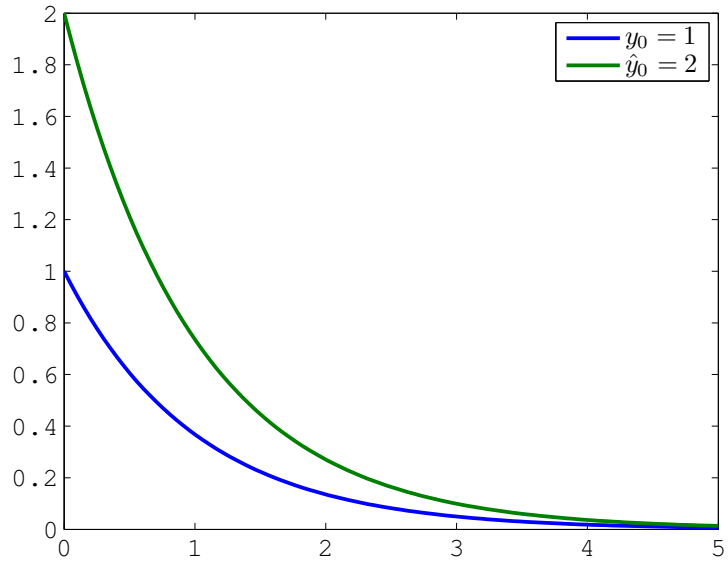
- We previously referred to this type of property as the conditioning of the problem
- Stability previously referred only to properties of a numerical approximation

In ODEs (and PDEs), it is standard to use stability to refer to sensitivity of both the mathematical problem and numerical approx.

ODE Stability

Consider stability of $y' = \lambda y$ (assuming $y(t) \in \mathbb{R}$) for different values of λ

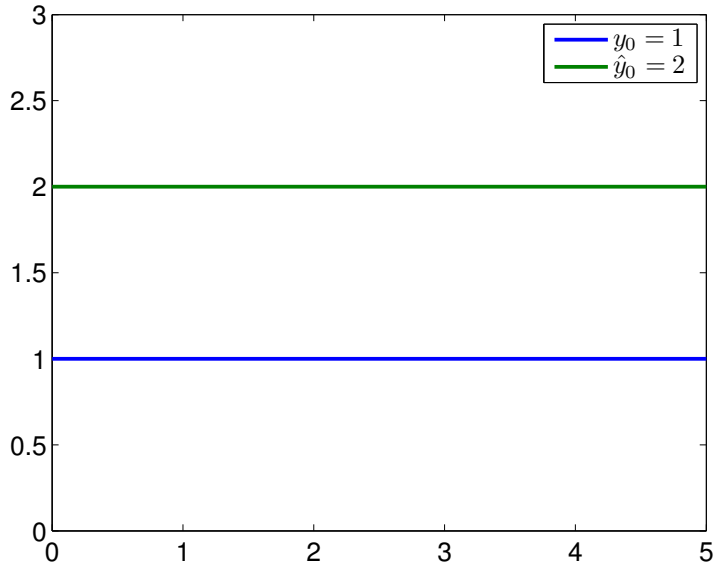
$$y(t) - \hat{y}(t) = (y_0 - \hat{y}_0)e^{\lambda t}$$



$\lambda = -1$, asymptotically stable

ODE Stability

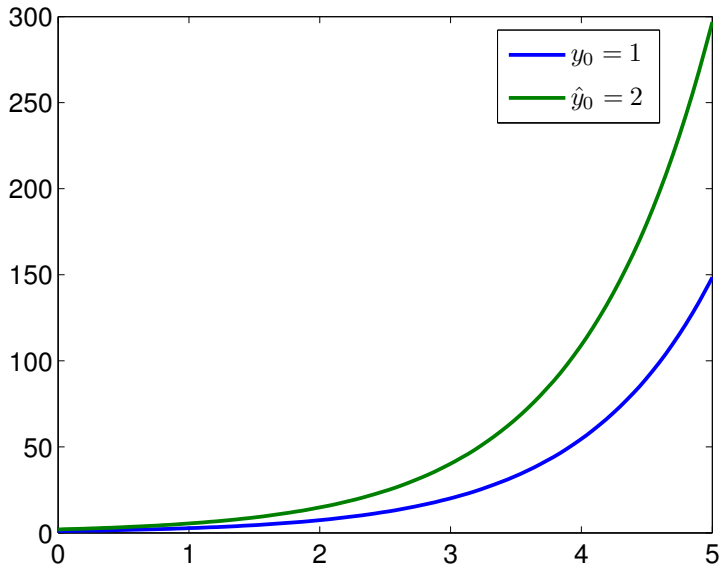
$$y(t) - \hat{y}(t) = (y_0 - \hat{y}_0)e^{\lambda t}$$



$\lambda = 0$, stable

ODE Stability

$$y(t) - \hat{y}(t) = (y_0 - \hat{y}_0)e^{\lambda t}$$



$\lambda = 1$, unstable

ODE Stability

More generally, we can allow λ to be a complex number: $\lambda = a + ib$

Then $y(t) = y_0 e^{(a+ib)t} = y_0 e^{at} e^{ibt} = y_0 e^{at}(\cos(bt) + i \sin(bt))$

The key issue for stability is now the sign of $a = \mathrm{Re}(\lambda)$:

- $\mathrm{Re}(\lambda) < 0 \implies$ asymptotically stable
- $\mathrm{Re}(\lambda) = 0 \implies$ stable
- $\mathrm{Re}(\lambda) > 0 \implies$ unstable

ODE Stability

Our understanding of the stability of $y' = \lambda y$ extends directly to the case $y' = Ay$, where $y \in \mathbb{R}^n, A \in \mathbb{R}^{n \times n}$

Suppose that A is diagonalizable, so that we have the eigenvalue decomposition $A = V \Lambda V^{-1}$, where

- $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, where the λ_j are eigenvalues
- V is matrix with eigenvectors as columns, $v_1, v_2, \ldots, v_n$

Then,

$$y' = Ay = V \Lambda V^{-1} y \implies V^{-1} y' = \Lambda V^{-1} y \implies z' = \Lambda z$$

where $z \equiv V^{-1} y$ and $z_0 \equiv V^{-1} y_0$

ODE Stability

Hence we have n decoupled ODEs for z, and stability of z_i is determined by λ_i for each i

Since z and y are related by the matrix V, then (roughly speaking) if all z_i are stable then all y_i will also be stable[5]

Hence assuming that V is well-conditioned, then we have:
$\mathrm{Re}(\lambda_i) \leq 0$ for $i = 1, \ldots, n \implies y' = Ay$ is a stable ODE

Next we consider stability of numerical approximations to ODEs

[5] "Roughly speaking" here because V can be ill-conditioned — a more precise statement is based on "pseudospectra", outside the scope of AM205

ODE Stability

Numerical approximation to an ODE is stable if:

For every $\epsilon > 0$, $\exists \delta > 0$ such that

$$\|\hat{y}_0 - y_0\| \leq \delta \implies \|\hat{y}_k - y_k\| \leq \epsilon$$

for all $k \geq 0$

Key idea: We want to develop numerical methods that mimic the stability properties of the exact solution

That is, if the ODE we're approximating is unstable, we can't expect the numerical approximation to be stable!

Stability

Since ODE stability is problem dependent, we need a standard "test problem" to consider

The standard test problem is the simple scalar ODE $y' = \lambda y$

Experience shows that the behavior of a discretization on this test problem gives a lot of insight into behavior in general

Ideally, to reproduce stability of the ODE $y' = \lambda y$, we want our discretization to be stable for all $\mathrm{Re}(\lambda) \leq 0$

Stability: Forward Euler

Consider forward Euler discretization of $y' = \lambda y$:

$$y_{k+1} = y_k + h\lambda y_k = (1 + h\lambda) y_k \implies y_k = (1 + h\lambda)^k y_0$$

Here $1 + h\lambda$ is called the amplification factor

Hence for stability, we require $|1 + \bar{h}| \leq 1$, where $\bar{h} \equiv h\lambda$

Let $\bar{h} = a + ib$, then $|1 + a + ib|^2 \leq 1^2 \implies (1 + a)^2 + b^2 \leq 1$

Stability: Forward Euler

Hence forward Euler is stable if $\bar{h} \in \mathbb{C}$ is inside the disc with radius 1, center $(-1, 0)$: This is a subset of "left-half plane," $\mathrm{Re}(\bar{h}) \leq 0$

As a result we say that the forward Euler method is conditionally stable: when $\mathrm{Re}(\lambda) \leq 0$ we have to restrict h to ensure stability

For example, given $\lambda \in \mathbb{R}_{<0}$, we require

$$-2 \leq h\lambda \leq 0 \implies h \leq -2/\lambda$$

Hence "larger negative λ" implies tighter restriction on h:

$$\begin{aligned} \lambda = -10 &\implies h \leq 0.2 \\ \lambda = -200 &\implies h \leq 0.01 \end{aligned}$$

Stability: Backward Euler

In comparison, consider backward Euler discretization for $y' = \lambda y$:

$$y_{k+1} = y_k + h\lambda y_{k+1} \implies y_k = \left(\frac{1}{1-h\lambda}\right)^k y_0$$

Here the amplification factor is $\frac{1}{1-h\lambda}$

Hence for stability, we require $\frac{1}{|1-h\lambda|} \leq 1$

Stability: Backward Euler

Again, let $\bar{h} \equiv h\lambda = a + ib$, we need $1^2 \leq |1 - (a + ib)|^2$, *i.e.* $(1 - a)^2 + b^2 \geq 1$

Hence, for $\mathrm{Re}(\lambda) \leq 0$, this is satisfied for any $h > 0$

As a result we say that the backward Euler method is unconditionally stable: no restriction on h for stability

Stability

Implicit methods generally have larger stability regions than explicit methods! Hence we can take larger timesteps with implicit

But explicit methods are require less work per time-step since don't need to solve for y_{k+1}

Therefore there is a tradeoff: The choice of method should depend on the details of the problem at hand

Runge–Kutta Methods

Runge–Kutta (RK) methods are another type of one-step discretization, a very popular choice

Aim to achieve higher order accuracy by combining evaluations of f (*i.e.* estimates of y') at several points in $[t_k, t_{k+1}]$

RK methods all fit within a general framework, which can be described in terms of Butcher tableaus

Instead of detailing this framework, we will just consider two RK examples: two evaluations of f and four evaluations of f

Runge–Kutta Methods

The family of Runge–Kutta methods with two intermediate evaluations is defined by

$$y_{k+1} = y_k + h(ak_1 + bk_2),$$

where $k_1 = f(t_k, y_k)$, $k_2 = f(t_k + \alpha h, y_k + \beta h k_1)$

The Euler method is a member of this family, with $a = 1$ and $b = 0$