AM213B Numerical Methods for the Solution of Differential Equations

Lecture 02

List of topics in this lecture

- Numerical methods for solving IVP of ODE, Euler method, backward Euler method, midpoint method, trapezoidal method
- Explicit vs implicit methods, single-step vs multi-step methods
- Operator notation, stability of a numerical method,
- Global error, local truncation error, error of one-step,
- Error analysis, relation between global error and local truncation error

---

Numerical methods for solving IVP of first order ODEs

$$\begin{cases} u' = f(u,t) \\ u(t_0) = u_0 \end{cases}$$

Discretization:

$t_n = t_0 + n\,h$ $\qquad$ h: time step

Strategy:

At t_n, the differential equation is

$$u'(t_n) = f\left(u(t_n), t_n\right)$$

We use a numerical differentiation to approximate the derivative

$$u'(t_n) \xleftarrow{\text{to replace}} \frac{u(t_n+h)-u(t_n)}{h}$$

$$\Longrightarrow \quad \frac{u(t_{n+1})-u(t_n)}{h} \approx f\left(u(t_n), t_n\right)$$

$$\Longrightarrow \quad u(t_{n+1}) \approx u(t_n) + h f\left(u(t_n), t_n\right)$$

In each time step, we use this approximate formula to calculate $u(t_{n+1})$ from $u(t_n)$.

We repeat this process over many time steps.

Notation:

u_n = numerical approximation of $u(t_n)$

With this notation, we write the numerical method as

Euler method:

$$u_{n+1} = u_n + h f(u_n, t_n)$$

- It is explicit. u_{n+1} is given explicitly in terms of u_n and function $f(x, t)$. There is no need to solve an equation to get u_{n+1}.
- It is a single-step method (also called "one-step method"). To calculate u_{n+1}, we only need to know u_n. There is no need to know u_{n-1}, u_{n-2}, …

An implicit method (backward Euler method):

At t_n, the differential equation is

$$u'(t_n) = f\left(u(t_n), t_n\right)$$

We use the backward numerical differentiation to approximate the derivative.

$$u'(t_n) \xleftarrow{\text{to replace}} \frac{u(t_n) - u(t_n - h)}{h}$$

$$\Longrightarrow \quad \frac{u(t_n) - u(t_{n-1})}{h} \approx f\left(u(t_n), t_n\right)$$

$$\Longrightarrow \quad u(t_n) \approx u(t_{n-1}) + h f\left(u(t_n), t_n\right)$$

We re-write it as

$$u(t_{n+1}) \approx u(t_n) + h f\left(u(t_{n+1}), t_n\right)$$

Backward Euler method:

$$u_{n+1} = u_n + h f\left(u_{n+1}, t_{n+1}\right)$$

- It is implicit. u_{n+1} is not given explicitly in terms of u_n. Rather u_{n+1} satisfies an equation. We need to solve this equation to get u_{n+1}.
- It is a single-step method.

A multi-step method (the midpoint method)

At t_n, the differential equation is

$$u'(t_n) = f\left(u(t_n), t_n\right)$$

We use the central difference to approximate the derivative

$$u'(t_n) \xleftarrow{\text{to replace}} \frac{u(t_n+h)-u(t_n-h)}{2h}$$

==> $$\frac{u(t_{n+1})-u(t_{n-1})}{2h} \approx f\left(u(t_n), t_n\right)$$

==> $$u(t_{n+1}) \approx u(t_{n-1}) + 2h f\left(u(t_n), t_n\right)$$

The midpoint method (also known as leap-frog method):

$$u_{n+1} = u_{n-1} + h f\left(u_n, t_n\right)$$

- It is explicit. u_{n+1} is given explicitly.
- It is a multi-step method. To calculate u_{n+1}, we need to know both u_n and u_{n-1}. To get started, we need both u_1 and u_0 to calculate u_2.

Forecast: we will see later in this lecture and in assignments that

- Euler method and backward Euler method are both first order.
- The midpoint method is second order but it does not work well.

To design a good second order method, we need a new strategy.

Integrating the differential equation from t_n to t_{n+1} yields

$$u'(t) = f\left(u(t), t\right)$$

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

==> $$u(t_{n+1}) - u(t_n) = \int_{t_n}^{t_{n+1}} f\left(u(t), t\right)dt$$

We use the trapezoidal rule to approximate the integral on the RHS.

Recall the trapezoidal rule: $$\int_{t_n}^{t_{n+1}} g(t)dt \approx \left(g(t_n) + g(t_{n+1})\right)\frac{h}{2}$$

We use the approximation

$$\int_{t_n}^{t_{n+1}} f\big(u(t),t\big)dt \xleftarrow{\text{to replace}} \Big(f\big(u(t_n),t_n\big)+f\big(u(t_{n+1}),t_{n+1}\big)\Big)\frac{h}{2}$$

==> $u(t_{n+1})-u(t_n)\approx\Big(f\big(u(t_n),t_n\big)+f\big(u(t_{n+1}),t_{n+1}\big)\Big)\frac{h}{2}$

==> $u(t_{n+1})\approx u(t_n)+\Big(f\big(u(t_n),t_n\big)+f\big(u(t_{n+1}),t_{n+1}\big)\Big)\frac{h}{2}$

The trapezoidal method:

$$u_{n+1}=u_n+\frac{h}{2}\big(f(u_n,t_n)+f(u_{n+1},t_{n+1})\big)$$

- It is implicit.
- It is a single-step method.

Remark:

Euler method and backward Euler method can also be derived Using this new strategy.

Using the approximation

$$\int_{t_n}^{t_{n+1}} f\big(u(t),t\big)dt \xleftarrow{\text{to replace}} f\big(u(t_n),t_n\big)h$$

==> Euler method

Using the approximation

$$\int_{t_n}^{t_{n+1}} f\big(u(t),t\big)dt \xleftarrow{\text{to replace}} f\big(u(t_{n+1}),t_{n+1}\big)h$$

==> Backward Euler method

Error analysis (convergence analysis)

Operator notation

To facilitate the discussion, we introduce the operator notation.

We write a numerical method in the form of an operator

$$u_{n+1}=L_{num}(u_n)$$

Examples:

Euler method:

The numerical operator L_{num} is

$$L_{num}(u_n) \equiv u_n + hf(u_n, t_n)$$

Backward Euler method:

We write the method as

$$\underbrace{u_{n+1} - hf(u_{n+1}, t_{n+1})}_{G(u_{n+1})} = u_n, \qquad G(u) \equiv u - hf(u, t_n + h)$$

The numerical operator L_{num} is

$$L_{num}(u_n) \equiv G^{(-1)}(u_n)$$

Trapezoidal method:

We write the method as

$$\underbrace{u_{n+1} - \frac{h}{2} f(u_{n+1}, t_{n+1})}_{G_2(u_{n+1})} = u_n + \frac{h}{2} f(u_n, t_n), \qquad G_2(u) \equiv u - \frac{h}{2} f(u, t_n + h)$$

The numerical operator L_{num} is

$$L_{num}(u_n) \equiv G_2^{(-1)}\left(u_n + \frac{h}{2} f(u_n, t_n)\right)$$

Remarks:

- The operator L_{num} depends on h
- The operator L_{num}, in general, also depends on t_{n}.

So the full notation for L_{num} should be $L_{\text{num}}(u_{\text{n}}, h, t_{\text{n}})$.

We write the numerical solution symbolically as

$$u_N = \left(L_{num}\right)^N (u_0)$$

where the power of L_{num} is interpreted as

$$\left(L_{num}\right)^N (u_0) = \cdots L_{num}\left(L_{num}\left(L_{num}(u_0, h, t_0), h, t_1\right), h, t_2\right)$$

Corresponding to the numerical operator L_{num}, we also introduce the exact solution operator L_{exa}

$$u(t_0 + T) = L_{exa}\left(u(t_0), T\right)$$

Stability of a numerical method

Definition:

A numerical method $u_{n+1} = L_{num}(u_n)$ is stable if there exists a constant C, independent of (h, u_n, v_n), such that for small h

$$\left|L_{num}(u_n) - L_{num}(v_n)\right| \le (1 + C \cdot h)\left|u_n - v_n\right| \qquad \text{for all } u_n \text{ and } v_n$$

Remarks:

- For a linear operator L_{num}, the stability is equivalent to

  $$\left|\left|L_{num}\right|\right| \le (1 + C \cdot h)$$

  But for solving non-linear ODEs, we have to use non-linear numerical methods.
- As we will see in the examples below and later, all reasonable single-step methods are stable provided that $f(u, t)$ is Lipschitz continuous.

Examples:

Suppose $f(u, t)$ is Lipschitz continuous.

Then Euler method, backward Euler method and trapezoidal method are all stable.

Euler method:

$$u_{n+1} = u_n + h f(u_n, t_n)$$
$$v_{n+1} = v_n + h f(v_n, t_n)$$

$$\Longrightarrow \quad \left|u_{n+1} - v_{n+1}\right| = \left|\left(u_n - v_n\right) + h\left(f(u_n, t_n) - f(v_n, t_n)\right)\right|$$
$$\le \left|u_n - v_n\right| + h\left|f(u_n, t_n) - f(v_n, t_n)\right|$$
$$\le \left(1 + hC_L\right)\left|u_n - v_n\right|$$

where C_L is the constant in the Lipschitz continuity.

Therefore, Euler method is stable.

Backward Euler method:

$$u_{n+1} - h f(u_{n+1}, t_{n+1}) = u_n$$
$$v_{n+1} - h f(v_{n+1}, t_{n+1}) = v_n$$

$$\Longrightarrow \quad \left|\left(u_{n+1} - v_{n+1}\right) - h\left(f(u_{n+1}, t_{n+1}) - f(v_{n+1}, t_{n+1})\right)\right| = \left|u_n - v_n\right|$$

$$LHS = \left|\left(u_{n+1} - v_{n+1}\right) - h\left(f(u_{n+1}, t_{n+1}) - f(v_{n+1}, t_{n+1})\right)\right|$$

$$\geq \left|u_{n+1}-v_{n+1}\right| - h\left|f(u_{n+1},t_{n+1})-f(v_{n+1},t_{n+1})\right|$$
$$\geq \left(1-hC_L\right)\left|u_{n+1}-v_{n+1}\right|$$

==> $\left(1-hC_L\right)\left|u_{n+1}-v_{n+1}\right| \leq \left|u_n-v_n\right|$

==> $\left|u_{n+1}-v_{n+1}\right| \leq \frac{1}{1-hC_L}\left|u_n-v_n\right|$

$$\leq \left(1+C\cdot h\right)\left|u_n-v_n\right| \quad \text{for small } h.$$

Therefore, backward Euler method is stable.

Trapezoidal method:

$$u_{n+1}-\frac{h}{2}f(u_{n+1},t_{n+1}) = u_n+\frac{h}{2}f(u_n,t_n)$$
$$v_{n+1}-\frac{h}{2}f(v_{n+1},t_{n+1}) = v_n+\frac{h}{2}f(v_n,t_n)$$

==> $\left|u_{n+1}-v_{n+1}-\frac{h}{2}\left(f(u_{n+1},t_{n+1})-f(v_{n+1},t_{n+1})\right)\right| = \left|u_n-v_n+\frac{h}{2}\left(f(u_n,t_n)-f(v_n,t_n)\right)\right|$

We examine the RHS and LHS separately.

$$RHS = \left|u_n-v_n+\frac{h}{2}\left(f(u_n,t_n)-f(v_n,t_n)\right)\right|$$

$$\leq \left|u_n-v_n\right| + \frac{h}{2}\left|f(u_n,t_n)-f(v_n,t_n)\right|$$
$$\leq \left(1+\frac{hC_L}{2}\right)\left|u_n-v_n\right|$$

$$LHS = \left|u_{n+1}-v_{n+1}-\frac{h}{2}\left(f(u_{n+1},t_{n+1})-f(v_{n+1},t_{n+1})\right)\right|$$

$$\geq \left|u_{n+1}-v_{n+1}\right| - \frac{h}{2}\left|f(u_{n+1},t_{n+1})-f(v_{n+1},t_{n+1})\right|$$
$$\geq \left(1-\frac{hC_L}{2}\right)\left|u_{n+1}-v_{n+1}\right|$$

Combine these two, we arrive at

$$\left(1-\frac{hC_L}{2}\right)\left|u_{n+1}-v_{n+1}\right| \leq LHS = RHS \leq \left(1+\frac{hC_L}{2}\right)\left|u_n-v_n\right|$$

$$\Longrightarrow \quad |u_{n+1} - v_{n+1}| \le \frac{1+\frac{hC_L}{2}}{1-\frac{hC_L}{2}} |u_n - v_n|$$

$$\le (1 + C \cdot h)|u_n - v_n| \quad \text{for small } h.$$

Therefore, backward Euler method is stable.

Next we introduce three types of errors.

- Global error, $E_N(h)$
- Local truncation error (LTE), $e_n(h)$
- Error of one step, $\tilde{e}_n(h)$

Global error

Recall the notation:

$u(t)$ = exact solution

u_n = numerical solution at t_n

We start the numerical solution at t_0 with $u_0 = u(t_0)$ (exact solution)

For a fixed value of $T = N\,h$, we compare u_N and $u(t_0+T)$.

u_N = numerical solution at $t_N = t_0 + N\,h = t_0 + T$

$u(t_0+T)$ = exact solution at $t_0 + T$

The global error is defined as

$$\underbrace{E_N(h)}_{\text{Global error}} \equiv \underbrace{u_N}_{\text{Numerical solution at } t_0+T} - \underbrace{u(t_0+T)}_{\text{Exact solution at } t_0+T}$$

In terms of the numerical operator, the global error is

$$E_N(h) \equiv \left(L_{num}\right)^N \left(u(t_0)\right) - u(t_N)$$

Remarks:

- The value of $T = N\,h$ is fixed as we vary h and N. For a smaller time step h, it takes more steps to reach the final time $T = N\,h$.
- The global error is what we want to know. But it is difficult to study directly.

  To study the global error, we first need to consider the local truncation error.

Local truncation error (LTE)

The local truncation error is defined as the residual term when substituting an exact solution into the numerical method.

In terms of the numerical operator, the local truncation error is

$$e_n(h) \equiv u\left(t_{n+1}\right) - L_{num}\left(u(t_n)\right)$$

where $u(t)$ is an exact solution.

Example: Local truncation errors of Euler, backward Euler, and Trapezoidal methods.

Euler method:

We write the method as

$$u_{n+1} - u_n - hf(u_n, t_n) = 0$$

The local truncation error of Euler method is

$$e_n(h) = u(t_{n+1}) - u(t_n) - hf\left(u(t_n), t_n\right)$$

We use Taylor expansion to study the local truncation error

$$u(t_{n+1}) = u(t_n + h) = u(t_n) + u'(t_n)h + u''(t_n)\frac{h^2}{2} + \cdots$$

Since $u(t)$ is an exact solution, we have

$$f\left(u(t_n), t_n\right) = u'(t_n)$$

Substituting these two into $e_n(h)$, we obtain

$$e_n(h) = u''(t_n)\frac{h^2}{2} + \cdots = O(h^2)$$

Backward Euler method:

We write the method as

$$u_{n+1} - u_n - hf(u_{n+1}, t_{n+1}) = 0$$

The local truncation error of backward Euler method is

$$e_n(h) = u(t_{n+1}) - u(t_n) - hf\left(u(t_{n+1}), t_{n+1}\right)$$

We expand $u(t_n)$ around t_{n+1}

$$u(t_n) = u(t_{n+1} - h) = u(t_{n+1}) - u'(t_{n+1})h + u''(t_{n+1})\frac{h^2}{2} + \cdots$$

Substituting it into $e_n(h)$ and using $f\left(u(t_{n+1}),t_{n+1}\right)=u'(t_{n+1})$, we get

$$e_n(h)=-u''(t_{n+1})\frac{h^2}{2}+\cdots=O(h^2)$$

Trapezoidal method:

We write the method as

$$u_{n+1}-u_n-\frac{h}{2}\left(f(u_n,t_n)+f(u_{n+1},t_{n+1})\right)=0$$

The local truncation error of Trapezoidal method is

$$e_n(h)=u(t_{n+1})-u(t_n)-\frac{h}{2}\left(f\left(u(t_n),t_n\right)+f\left(u(t_{n+1}),t_{n+1}\right)\right)$$

We expand everything around t_n.

$$u(t_{n+1})-u(t_n)=u(t_n+h)-u(t_n)$$

$$=u'(t_n)h+u''(t_n)\frac{h^2}{2}+u^{(3)}(t_n)\frac{h^3}{3!}+\cdots$$

$$f\left(u(t_n),t_n\right)+f\left(u(t_{n+1}),t_{n+1}\right)=u'(t_n)+u'(t_{n+1})$$

$$=2u'(t_n)+u''(t_n)h+u^{(3)}(t_n)\frac{h^2}{2}+\cdots$$

Substituting these into $e_n(h)$, we have

$$e_n(h)=-u^{(3)}(t_n)\frac{h^3}{12}+\cdots=O(h^3)$$

Remarks:

- We use Taylor expansion to study the local truncation error.
- The definition of local truncation error has some ambiguity. We can see that in the example below.

Example (ambiguity in the definition of local truncation error):

Backward Euler method:

Version1: $u_{n+1}=u_n+hf(u_{n+1},t_{n+1})$

Version 2: $u_{n+1} = G^{(-1)}\left(u_n\right), \qquad G(u) \equiv u - hf\left(u, t_n + h\right)$

Local truncation error of Version 1:

$$e_n^{(V1)}(h) = u(t_{n+1}) - hf\left(u(t_{n+1}), t_{n+1}\right) - u(t_n)$$

Local truncation error of Version 2:

$$e_n^{(V2)}(h) = u(t_{n+1}) - G^{(-1)}\left(u(t_n)\right)$$

Comparing $e_n^{(V1)}(h)$ and $e_n^{(V2)}(h)$, we notice that

$$e_n^{(V1)}(h) = \underbrace{u(t_{n+1}) - hf\left(u(t_{n+1}), t_{n+1}\right)}_{G\left(u(t_{n+1})\right)} - u(t_n) = G\left(u(t_{n+1})\right) - u(t_n)$$

$$= G\left(u(t_{n+1})\right) - G\left(G^{(-1)}\left(u(t_n)\right)\right)$$

using the mean value theorem, we write it as

$$= \frac{\partial G(u^*)}{\partial u}\left[u(t_{n+1}) - G^{(-1)}\left(u(t_n)\right)\right] = \frac{\partial G(u^*)}{\partial u} e_n^{(V2)}(h)$$

The derivative of $G(u) \equiv u - hf\left(u, t_n + h\right)$ is

$$\frac{\partial G(u^*)}{\partial u} = 1 - h\frac{\partial f(u^*, t_n + h)}{\partial u}$$

Thus, $e_n^{(V1)}(h)$ and $e_n^{(V2)}(h)$ are related by

$$e_n^{(V1)}(h) = \left(1 - h\frac{\partial f(u^*, t_n + h)}{\partial u}\right) e_n^{(V2)}(h)$$

$e_n^{(V1)}(h)$ and $e_n^{(V2)}(h)$ are different. But they have the same leading term.

In conclusion, it does not matter which version we use.

To relate the global error to the local truncation error, we introduce error of one step.

Error of one step

Start the numerical solution at t_n with $u_n = u(t_n)$ (the exact solution at t_n).

Go forward ONE step and then compare u_{n+1} and $u(t_{n+1})$

u_{n+1} = numerical solution at t_{n+1} starting at t_n with $u_n = u(t_n)$

$u(t_{n+1})$ = exact solution at t_{n+1}

The error of one step is defined as

$$\underbrace{\tilde{e}_n(h)}_{\text{Error of one step}} \equiv \underbrace{u_{n+1}}_{\substack{\text{Numerical solution starting at } t_n \text{ with exact solution}}} - \underbrace{u(t_{n+1})}_{\text{Exact solution}}$$

In terms of the numerical operator, the error of one step is

$$\tilde{e}_n(h) \equiv L_{num}\left(u(t_n)\right) - u(t_{n+1})$$

Comparing with $e_n(h) \equiv u(t_{n+1}) - L_{num}\left(u(t_n)\right)$, we conclude

$$\tilde{e}_n(h) = -e_n(h)$$

Remark:

- In absolute value, the error of one step is essentially the same as the local truncation error; they have the same leading term.

  Therefore, from now on we will use $e_n(h)$ to denote both the local truncation error and the error of one step without distinguishing between these two.

Relation between global error and local truncation error

Theorem:

Suppose the numerical method is stable. Then the global error $E_N(h)$ is one order lower than the local truncation error $e_n(h)$. In other words, if $e_n(h) = O(h^{p+1})$, then $E_N(h) = O(h^p)$.

Proof:

We need to show $\left|E_N(h)\right| \le C \cdot h^p$. Recall the definition

$$E_N(h) = u_N - u(t_N).$$

We derive a recursive inequality connecting $E_{n+1}(h)$ to $E_n(h)$.

We write $E_{n+1}(h)$ as the sum of two parts.

$$\begin{aligned}\left|E_{n+1}(h)\right| &= \left|u_{n+1} - u(t_{n+1})\right| \\ &= \left|u_{n+1} - L_{num}\left(u(t_n)\right) + L_{num}\left(u(t_n)\right) - u(t_{n+1})\right| \\ &\le \left|u_{n+1} - L_{num}\left(u(t_n)\right)\right| + \left|L_{num}\left(u(t_n)\right) - u(t_{n+1})\right|\end{aligned}$$

$$= \underbrace{\left|L_{num}(u_n) - L_{num}(u(t_n))\right|}_{\substack{u_n \text{ and } u(t_n), \text{ each} \\ \text{propagated by one step}}} + \underbrace{\left|L_{num}(u(t_n)) - u(t_{n+1})\right|}_{\text{Error of one step } e_n(h)}$$

Using the stability of numerical method and $e_n(h) = O(h^{p+1})$, we get

$$\left|E_{n+1}(h)\right| \le (1 + C_1 h)\left|\underbrace{u_n - u(t_n)}_{E_n(h)}\right| + C_2 h^{p+1}$$

Thus, we obtain a recursive inequality connecting $E_{n+1}(h)$ to $E_n(h)$.

$$\boxed{\begin{aligned} &\left|E_{n+1}(h)\right| \le (1 + C_1 h)\left|E_n(h)\right| + C_2 h^{p+1} \\ &E_0(h) = 0 \end{aligned}} \qquad \text{(E01)}$$

To solve this recursive inequality, we use a method similar to "the integrating factor method" for solving ODEs.

Multiplying both sides by $(1+C_1h)^{-(n+1)}$ and moving E_n to left, we re-write it as

$$(1 + C_1 h)^{-(n+1)}\left|E_{n+1}(h)\right| - (1 + C_1 h)^{-n}\left|E_n(h)\right| \le C_2 h^{p+1}(1 + C_1 h)^{-(n+1)}$$

Summing from $n = 0$ to $n = N - 1$ and using $E_0 = 0$, we get

$$\begin{aligned} (1 + C_1 h)^{-N}\left|E_N(h)\right| &\le C_2 h^{p+1} \sum_{n=0}^{N-1} (1 + C_1 h)^{-(n+1)} \\ &\le C_2 h^{p+1} \cdot (1 + C_1 h)^{-1} \frac{1 - (1 + C_1 h)^{-N}}{1 - (1 + C_1 h)^{-1}} \\ &= \frac{C_2}{C_1} h^p \left(1 - (1 + C_1 h)^{-N}\right) \end{aligned}$$

In the above, we have used the identity

$$\sum_{n=0}^{N-1} r^{n+1} = r \frac{1 - r^N}{1 - r}$$

Multiplying by $(1+C_1h)^N$ and using $1 + C_1 h \le e^{C_1 h}$, we arrive at

$$\left|E_N(h)\right| \le \frac{C_2}{C_1} h^p \left((1 + C_1 h)^N - 1\right) \le \frac{C_2}{C_1} h^p \left(e^{C_1 hN} - 1\right) = \frac{C_2}{C_1}\left(e^{C_1 T} - 1\right) h^p \quad \text{where } T = N h$$

Therefore, the global error is bounded by

$$\left|E_N(h)\right| \le C_2 \frac{e^{C_1 T} - 1}{C_1} h^p = O(h^p)$$