Lecture 12: Error in IVPs

Today:

- Discuss ***error*** in finite-difference methods for IVPs
  - Characterize ***truncation error***
  - Characterize ***stability***

Characterizing the error of one-step and multi-step methods

We previously derived one-step and multi-step finite-difference methods for solving IVPs.

Let's now transition to an important question: what is the error associated with these methods?

Global error

The *global error* of a finite-difference method at some time instance t_k is defined as $\boldsymbol{e}_k := \boldsymbol{u}(t_k) - \boldsymbol{u}_k$. Often, this error is expressed succinctly using an appropriately defined norm $||\cdot||$ as $||\boldsymbol{e}_k||$.

There are two contributions to the global error: the truncation error advancing from one time step to the next, and the accumulated error over all past time steps.

The ***truncation error*** associated with advancing from one time step to the next

The ***accumulated error*** over all past time steps.

We will build intuition for these, and then consider the ***truncation error*** in more detail.

Truncation Error

Building intuition for the sources of error with the Forward Euler method

To see what the two sources of error are, consider the FE method.

The ***truncation error*** is the error associated with applying the numerical method to the ***true solution***

We're using the ***true solution*** here, not u_k, u_{k+1}

FE method

$$\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - f(u(t_k), t_k)$$

$$\mathbf{u}_{k+1} - \mathbf{u}_k = \Delta t \mathbf{f}(\mathbf{u}_k, t_k)$$

This error comes from the fact that even if we started with the perfect solution at $u(t_k)$, we would incur some error in getting our approximation at $t = t_{k+1}$

The ***accumulation error*** is the collection of the truncation errors over all previous time steps, not just in going from t_k to t_{k+1}

Let's now consider how to quantify the truncation error for ***one-step*** and ***multi-step*** methods

Quantifying the truncation error (TE) for one-step methods

Let's quantify the TE for the FE method first

$$\boldsymbol{\tau}_k = \frac{\boldsymbol{u}(t_{k+1}) - \boldsymbol{u}(t_k)}{\Delta t} - \boldsymbol{f}(\boldsymbol{u}(t_k), t_k)$$

$$= \frac{\boldsymbol{u}(t_{k+1}) - \boldsymbol{u}(t_k)}{\Delta t} - \dot{\boldsymbol{u}}(t_k) \quad \text{[using the definition of an IVP]}$$

$$= \frac{\boldsymbol{u}(t_{k+1}) - \boldsymbol{u}(t_k)}{\Delta t} - \frac{\boldsymbol{u}(t_{k+1}) - \boldsymbol{u}(t_k)}{\Delta t} + O(\Delta t)$$

[using a Taylor series expansion of $\boldsymbol{u}(t_k)$ about t_{k+1}]

$$\implies \quad \boldsymbol{\tau}_k = O(\Delta t)$$

$$\mathbf{u}(t_{k+1}) = \mathbf{u}(t_k) + \dot{\mathbf{u}}(t_k)\frac{(t_{k+1} - t_k)}{1!} + \ddot{\mathbf{u}}(t_k)\frac{(t_{k+1} - t_k)^2}{2!} + \ldots$$

$$= \mathbf{u}(t_k) + \dot{\mathbf{u}}(t_k)\Delta t + \ddot{\mathbf{u}}(t_k)\frac{(\Delta t)^2}{2} + \ldots$$

$$\implies \dot{\mathbf{u}}(t_k) = \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} + \ddot{\mathbf{u}}(t_k)\frac{\Delta t}{2} + \ldots$$

$$= \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} + O(\Delta t)$$

Big O notation

$\ddot{\mathbf{u}}(t_k)\frac{\Delta t}{2} + \ldots \leq M\Delta t$ as $\Delta t \to 0$

where M is some constant

An exercise with truncation error of one-step methods

Exercise. Write out the TE for the Backward Euler method and Heun's method

$$\tau_k = \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} - \mathbf{f}(\mathbf{u}(t_{k+1}), t_{k+1})$$ ***BE***

$$\tau_k = \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} - \frac{1}{2}\Big[\mathbf{f}(\mathbf{u}(t_k), t_k) + \mathbf{f}\big(\mathbf{u}(t_k) + \Delta t\,\mathbf{f}(\mathbf{u}(t_k), t_k), t_k\big)\Big]$$ ***Heun's***

Exercise. Quantify the TE for the Backward Euler method

$$\tau_k = \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} - \mathbf{f}(\mathbf{u}(t_{k+1}), t_{k+1})$$

Notice that $\mathbf{u}(t_k) = \mathbf{u}(t_{k+1}) - \dot{\mathbf{u}}(t_{k+1})\Delta t + \ddot{\mathbf{u}}(t_{k+1})\frac{\Delta t^2}{2} + \ldots$

$$\implies \dot{\mathbf{u}}(t_{k+1}) = \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} + \ddot{\mathbf{u}}(t_{k+1})\frac{\Delta t}{2} + \ldots$$

$$\tau_k = \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} - \dot{\mathbf{u}}(t_{k+1})$$

$$\tau_k = \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} - \frac{\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k)}{\Delta t} - \ddot{\mathbf{u}}(t_{k+1})\frac{\Delta t}{2} + \ldots$$

$$\implies \tau_k = O(\Delta t)$$

Constructing and quantifying TE for multi-step methods

Whereas for one-step methods the TE has to be constructed on a case by case basis, the procedure is more generic for multi-step methods…

Truncation error: multi-step methods

An r-step method defined using (9) has a truncation error given by

$$\boldsymbol{\tau}_k = \frac{1}{\Delta t}\left[\sum_{j=k-r+1}^{k+1} \alpha_{j-(k-r+1)}\boldsymbol{u}(t_j) - \Delta t \sum_{j=k-r+1}^{k+1} \beta_{j-(k-r+1)}\boldsymbol{f}\Big(\boldsymbol{u}(t_j), t_j\Big)\right] \tag{12}$$

The truncation error can be quantified through different Taylor series (see the typed notes):

If this = 0… and this = 0 $\implies \tau_k = O(\Delta t)$

$$\boldsymbol{\tau}_k = \frac{1}{\Delta t}\left(\sum_{j=k-r+1}^{k+1} \alpha_{j-(k-r+1)}\right)\boldsymbol{u}(t_k) + \left(\sum_{j=k-r+1}^{k+1}\left[(j-k)\alpha_{j-(k-r+1)} - \beta_{j-(k-r+1)}\right]\right)\dot{\boldsymbol{u}}(t_k) +$$
$$\Delta t\left(\sum_{j=k-r+1}^{k+1}\left[\frac{1}{2}(j-k)^2\alpha_{j-(k-r+1)} - (j-k)\beta_{j-(k-r+1)}\right]\right)\ddot{\boldsymbol{u}}(t_k) + \cdots + \tag{*}$$
$$\Delta t^{q-1}\left(\sum_{j=k-r+1}^{k+1}\left[\frac{1}{q!}(j-k)^q\alpha_{j-(k-r+1)} - \frac{1}{(q-1)!}(j-k)^{q-1}\beta_{j-(k-r+1)}\right]\right)\frac{d^q\boldsymbol{u}}{dt^q}\bigg|_{t_k}$$

and this = 0 $\implies \tau_k = O(\Delta t^2)$

Summary: truncation error for multi-step methods

Steps for establishing the truncation error of a multi-step method:

(A) The multi-step method will be given to you. From that, figure out the α and β coefficients.

(B) Check to see what conditions these coefficients satisfy, and use equation (*) to see if the method has a truncation error that is $O(\Delta t)$, $O(\Delta t^2)$, etc.

Accumulated Error

But what about global error?

How do we relate the ***truncation error*** to the ***global error*** that we actually care about?

We introduce the concept of ***stability***…

Remember that the second contribution to the ***global error*** was the ***accumulated error*** that accrues over the past k time steps.

We will define a notion of stability that ensures that this error doesn't grow out of hand.

Once we have that, we will be able to say that:

A finite difference method for an IVP will ***converge*** to the true solution (i.e., the FD solution will get infinitesimally close to the true solution as $\Delta t \to 0$) if

(A) The truncation error satisfies $\tau_k = O(\Delta t^p)$ for (an integer) $p \geq 1$
(B) The method is stable (we will define this concept later) at $\Delta t = 0$.

We call a FD method satisfying these properties
"order p accurate"

Building intuition for ***absolute stability*** through a model IVP

Let's start to build intuition for our notion of stability by considering the model problem for stability

$$\dot{\boldsymbol{u}} = \boldsymbol{\Lambda u}$$

$$\boldsymbol{u}(t_0) = \boldsymbol{u}_0$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & \cdots & \lambda_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & \lambda_n \end{bmatrix}$$

Why do we allow complex values of λ? See the side bar of the typed notes for details!

with each $\lambda_l \in \mathbb{C}$, $l = 1, \ldots, n$

It turns out that the exact solution to this problem is

jth entry of $\mathbf{u}_0$

This solution will decay when $Real(\lambda) < 0$ or blow up when $Real(\lambda) > 0$. Frequency of oscillations is given by $Imaginary(\lambda)$

$$u_j(t) = e^{\lambda_j(t-t_0)}(u_0)_j$$

How will we use this model problem to understand stability?? We will first define stability for one-step methods, then look at multi-step methods. Let's consider applying FE to the problem first.

Building intuition for ***absolute stability:*** applying FE to the model IVP

$$\begin{aligned}\boldsymbol{u}_{k+1} &= \boldsymbol{u}_k + \Delta t \boldsymbol{\Lambda} \boldsymbol{u}_k \\ &= (\boldsymbol{I} + \Delta t \boldsymbol{\Lambda})\boldsymbol{u}_k \\ &= (\boldsymbol{I} + \Delta t \boldsymbol{\Lambda})(\boldsymbol{I} + \Delta t \boldsymbol{\Lambda})\boldsymbol{u}_{k-1} \\ &= (\boldsymbol{I} + \Delta t \boldsymbol{\Lambda})^{k+1}\boldsymbol{u}_0\end{aligned}$$

Or, looking at the j^{th} entry specifically:

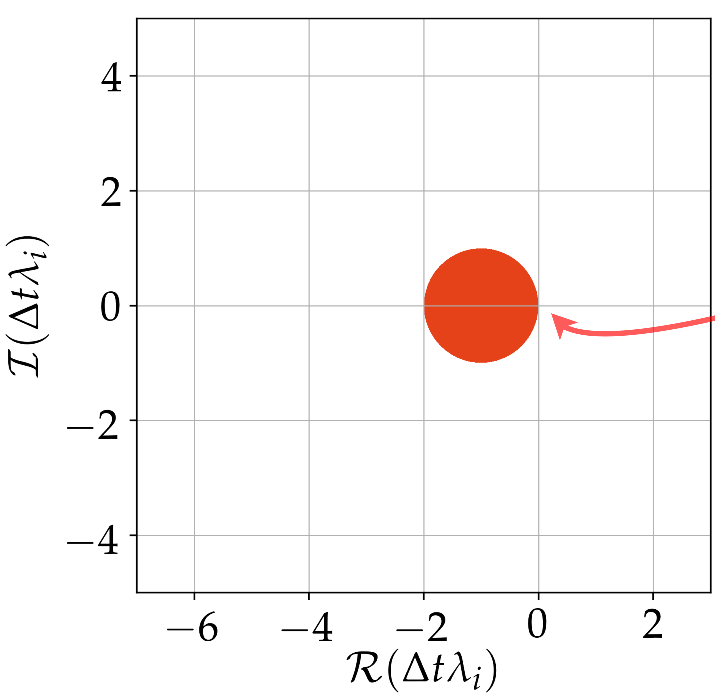
$$(u_{k+1})_j = (1 + \Delta t \lambda_j)^{k+1}(u_0)_j$$

What does this mean?

If $|1 + \Delta t \lambda_j| < 1$, then $(u_{k+1})_j$ will eventually $\rightarrow 0$ when k becomes large enough

If $|1 + \Delta t \lambda_j| > 1$, then $(u_{k+1})_j$ will eventually $\rightarrow \infty$ when k becomes large enough

Gives a criteria for identifying stability! Our method is ***absolutely stable*** if $|1 + \Delta t \lambda_j| < 1$



Stable if $\Delta t \lambda$ is in the orange circle

Verify this for yourselves…
Recall that $|z| = |x + yi| = \sqrt{(x^2 + y^2)}$

Your turn: determine the absolute stability criteria for the Backward Euler method

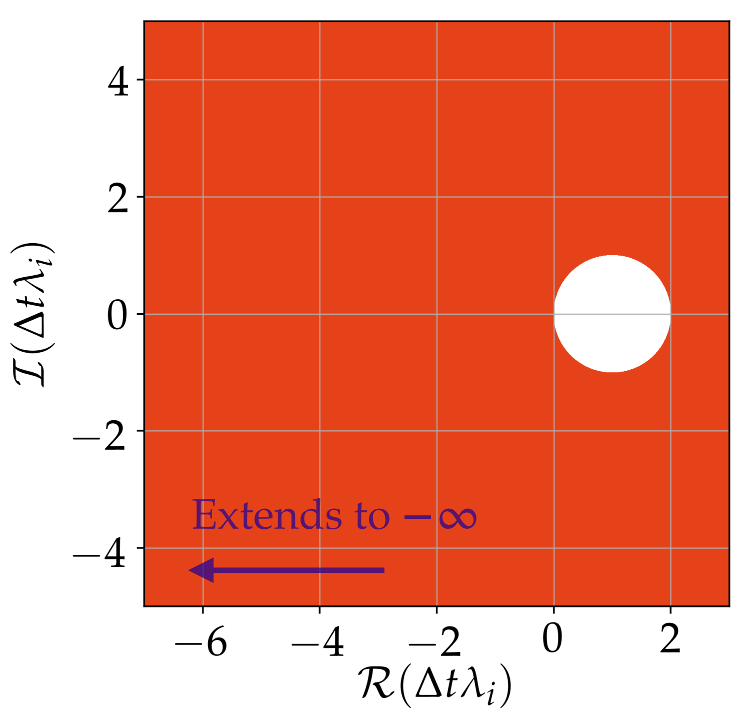
$$\begin{aligned}\boldsymbol{u}_{k+1} &= \boldsymbol{u}_k + \Delta t \boldsymbol{\Lambda} \boldsymbol{u}_{k+1} \\ &= (\boldsymbol{I} - \Delta t \boldsymbol{\Lambda})^{-1} \boldsymbol{u}_k \\ &= [(\boldsymbol{I} - \Delta t \boldsymbol{\Lambda})^{-1}]^{k+1} \boldsymbol{u}_0\end{aligned}$$

$$(u_{k+1})_j = \frac{1}{(1 - \Delta t \lambda_j)^{k+1}} (u_0)_j$$

If $1/|1 - \Delta t \lambda_j| < 1$, then $(u_{k+1})_j$ will eventually $\rightarrow 0$ when k becomes large enough

If $1/|1 - \Delta t \lambda_j| > 1$, then $(u_{k+1})_j$ will eventually $\rightarrow \infty$ when k becomes large enough

Our method is ***absolutely stable*** if $1/|1 - \Delta t \lambda_j| < 1$



The solutions won't blow up to infinity for a *much* wider range of $\Delta t \lambda_j$

Doesn't ensure accuracy! Just means the solutions won't grow infinitely large

See the typed notes for yet another example involving RK4

General approach to absolute stability for one-step methods

Notice that ***both*** FE and BE led to a relationship between $(u_{k+1})_j$ and $(u_0)_j$ of the form

$$(u_{k+1})_j = R^{k+1}(w)(u_0)_j$$

$w = \lambda_j \Delta t$

e.g., for FE

$$(u_{k+1})_j = \underbrace{(1 + \Delta t \lambda_j)}_{R(w)}{}^{k+1}(u_0)_j$$

It turns out this is generally true for one-step methods. So to determine absolute stability:

(A) Establish the relationship between $(u_{k+1})_j$ and $(u_0)_j$ to determine $R(w)$

(B) Find the values of w for which $|R(w)| < 1$ (the typed notes gives some Matlab code for how to do this)

Punchline: a one-step method is absolutely stable for the w values for which $|R(w)| < 1$

Absolute stability for multi-step methods

If we apply our general formula for a multi-step method to our model problem:

$$\sum_{j=k-r+1}^{k+1} \alpha_{j-(k-r+1)} \boldsymbol{u}_j = \Delta t \sum_{j=k-r+1}^{k+1} \beta_{j-(k-r+1)} \boldsymbol{\Lambda} \boldsymbol{u}_j$$

$$\implies \qquad \sum_{j=k-r+1}^{k+1} \left[\alpha_{j-(k-r+1)} \boldsymbol{I} - \Delta t \beta_{j-(k-r+1)} \boldsymbol{\Lambda} \right] \boldsymbol{u}_j = 0$$

Or for the l^{th} component

$$\sum_{j=k-r+1}^{k+1} \left[\alpha_{j-(k-r+1)} - \Delta t \beta_{j-(k-r+1)} \lambda_l \right] (u_j)_l = 0 \qquad (*)$$

Now here's the tricky part: we will ***assume*** that solutions to (*) can be expressed as polynomials. That is, we will replace $(u_j)_l$ with ζ^{j+r-1} in (*):

$$\sum_{j=k-r+1}^{k+1} \left[\alpha_{j-(k-r+1)} - \Delta t \beta_{j-(k-r+1)} \lambda_l \right] \zeta^{j+r-1} = 0$$

Clean up notation: divide both sides by ζ^k and rework indexing:

$$(**) \qquad \sum_{j=0}^{r} \left[\alpha_j - \Delta t \beta_j \lambda_l \right] \zeta^j = 0$$

Punchline: solutions to the model problem $\dot{\mathbf{u}} = \boldsymbol{\Lambda} \mathbf{u}$ are given by the roots of this equation.

Absolute stability for multi-step methods (cont)

(**) $$\sum_{j=0}^{r} \left[\alpha_j - \Delta t \beta_j \lambda_l\right] \zeta^j = 0$$

Punchline: solutions to the model problem $\dot{\mathbf{u}} = \Lambda \mathbf{u}$ are given by the roots of this equation.

Call the roots of (**) $\zeta_1, \zeta_2, \ldots, \zeta_r$

Synthesize. What does this mean? Work backwards:

- If we have $\zeta_1, \zeta_2, \ldots, \zeta_r$ that solve (**), then we can write

$$(u_j)_l = \sum_{m=1}^{r} c_m \zeta_m^j$$

and that $(u_j)_l$ ***will*** solve (*)

- Now let's say any one of the roots, call it ζ_g, has an absolute value > 1
- Then advancing $(u_j)_l$ in time means that as j gets larger, ζ_g^j will grow to infinity as j gets larger and larger

$\implies (u_j)_l$ will grow to infinity!

- Gives us a criteria for stability of multi-step methods!

For a multi-step method to be stable, each of the $\zeta_1, \zeta_2, \ldots, \zeta_r$ must have absolute value < 1

Absolute stability for multi-step methods (cont)

Let's make this stability criterion more precise:

This is the equation number in the typed notes.
It is equation (**) in our slides

Absolute stability criterion: multi-step methods

An r-step method is called *absolutely stable* for values of $\Delta t \lambda_l$ that yield solutions $\zeta_1, \ldots, \zeta_r$ to (20) satisfying $|\zeta_1| < 1$, $|\zeta_2| < 1, \ldots, |\zeta_r| < 1$. The method is *unstable* for values of $\Delta t \lambda_l$ that do not satisfy that criteria.

Should be eqn (29) in notes!

So what's the recipe for determining the region of absolute stability for multi-step methods?

(A) Determine the α, β coefficients for the multi-step method of interest

(B) Build the polynomial equation (**) and solve for the roots $\zeta_1, \zeta_2, \ldots, \zeta_r$ in terms of $\Delta t \lambda_l$

(C) Figure out the values of $\Delta t \lambda_l$ for which ***all*** roots are < 1

Let's consider an example to try to make this more tangible

An example of absolute stability for multi-step methods

Consider AB2.

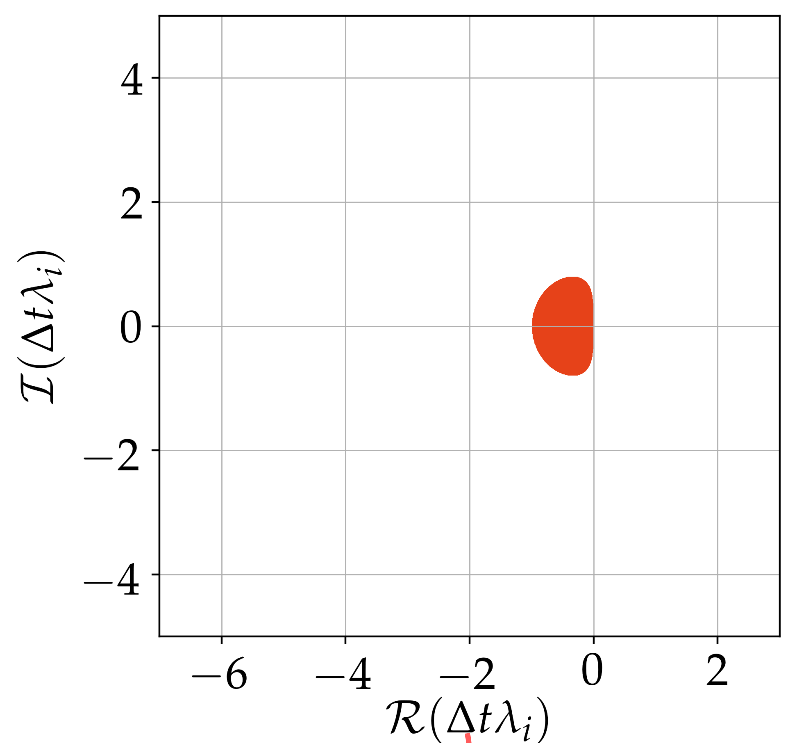
- We said last week that the $\alpha,\ \beta$ coefficients for this method are

$$\alpha_0 = 0,\ \alpha_1 = -1, \alpha_2 = 1$$
$$\beta_0 = -\frac{1}{2}, \beta_1 = \frac{3}{2}, \beta_2 = 0$$

- Plugging these into (**) for $r = 2$ gives

$$\Big[\alpha_0 - (\Delta t\lambda_l)\beta_0\Big] + \Big[\alpha_1 - (\Delta t\lambda_l)\beta_1\Big]\zeta + \Big[\alpha_2 - (\Delta t\lambda_l)\beta_2\Big]\zeta^2 = 0$$

$$\implies \quad \Big[(\Delta t\lambda_l)\frac{1}{2}\Big] + \Big[-1 - (\Delta t\lambda_l)\frac{3}{2}\Big]\zeta + \Big[1\Big]\zeta^2 = 0$$



- Can solve for ζ to get

$$\zeta = \frac{[1 + (\Delta t\lambda_l)\frac{3}{2}] \pm \sqrt{[1 + (\Delta t\lambda_l)\frac{3}{2}]^2 - 4\left[(\Delta t\lambda_l)\frac{1}{2}\right]}}{2}$$

- Evaluate this for many different $\Delta t\lambda_l$ values and identify where $|\zeta| < 1$