Lecture 10: Initial Value Problem

Today:

Based on local polynomial interpolation!

- *New topic:* **Initial value problems**
- Introduce numerical solution procedure: ***finite difference methods***
  - Describe a class of methods called ***one-step methods***

Where are we up to now?

We can now find a function, $f_a(x)$, that approximates a given function, $f(x)$, *accurately* on $x \in [a, b]$

Next few weeks: use these techniques to numerically approximate solutions to ***initial value problems (IVPs):***

$\frac{d\mathbf{u}}{dt}$

$$\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u}, t) \quad (1)$$

$$\mathbf{u}(t_0) = \mathbf{u}_0 \quad (2)$$

We do NOT know $\mathbf{u}$

We DO know $\mathbf{f}(\mathbf{u}, t)$ and $\mathbf{u}(t_0)$

Reminder: what is an IVP?

Whereas BVPs describe the *static (equilibrium)* response of a system to external forcing, IVPs govern the *dynamical time evolution* of a system to excitation.

The solution strategy we will use is called a ***finite difference method:***

Premise: For an integer $k = 0, 1, \ldots$

k is the ***time step index***

t is the ***actual time***

Problem statement: finite-difference methods for IVPs

Given: $\boldsymbol{u}_k \approx \boldsymbol{u}(t_k) = \boldsymbol{u}(t_0 + k\Delta t)$

Compute: $\boldsymbol{u}_{k+1} \approx \boldsymbol{u}(t_{k+1}) = \boldsymbol{u}(t_k + \Delta t)$.

$\mathbf{u}_k$ is our ***approximation*** at $t = t_k$

$\mathbf{u}(t_k)$ is the ***true value*** at $t = t_k$

How do we approximate $\mathbf{u}(t_{k+1})$?

Notice that we can integrate the IVP eqn (1) from t_k to t_{k+1}:

$$\int_{t_k}^{t_{k+1}} \dot{\mathbf{u}}\, dt = \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{u}, t)\, dt$$

$$\implies \mathbf{u}(t_{k+1}) - \mathbf{u}(t_k) = \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{u}, t)\, dt \qquad (3)$$

[Fundamental theorem of calculus]

The resulting methods are called ***finite difference methods***

But what about the RHS? Interpolate $\mathbf{f}$!

We will ***only*** consider local polynomial interpolation of $\mathbf{f}$ in this class

Today: we will explore some ***one step methods***

Methods that use ***only*** information over the interval $[t_k, t_{k+1}]$ to interpolate $\mathbf{f}$

The simplest one step finite difference method: Forward Euler

Let's consider an example together.

Let's interpolate $\mathbf{f}$ on $t \in [t_k, t_{k+1}]$ onto $\mathscr{P}^0[t_k, t_{k+1}]$:

$$\mathbf{f}_a(\mathbf{u}, t) = \mathbf{f}(\mathbf{u}(t_k), t_k)$$

Treat $\mathbf{f}$ as a constant over $t \in [t_k, t_{k+1}]$, with constant value given by evaluating $\mathbf{f}$ at t_k

Could evaluate at other times as well!

Then (3) becomes

$$\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k) = \mathbf{f}(\mathbf{u}(t_k), t_k) \int_{t_k}^{t_{k+1}} dt$$

$$\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k) = \mathbf{f}(\mathbf{u}(t_k), t_k)\Delta t \qquad (4)$$

$\Delta t = t_{k+1} - t_k$

This suggests a formula for finding the approximation to $\mathbf{u}(t_{k+1})$ given $\mathbf{u}(t_k)$:

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta t \mathbf{f}(\mathbf{u}_k, t_k) \qquad (5)$$

Note. In (4) we use $\mathbf{u}(t_k)$ whereas in (5) we use $\mathbf{u}_k$

The ***real*** $\mathbf{u}$ at $t = t_k$

Our ***approximation*** to the ***real*** $\mathbf{u}$ at $t = t_k$

An example involving the forward Euler method

Consider the IVP

$$\dot{u} = -3u \qquad (6)$$

$$u(t=0) = 4 \qquad (7)$$

Develop a strategy for approximating a solution forward in time using the Forward Euler method.

By equation (5), $u_{k+1} = u_k + \Delta t f(u_k, t_k) = u_k - 3(\Delta t) u_k$

So we would start at $t = 0$ and pick our small time step Δt. Then our approximation to the solution at Δt is

$$\begin{aligned} u_{\Delta t} &= u_0 - 3u_0 \Delta t \\ &= 4 - 3(4)\Delta t \end{aligned}$$

Once we have that we could advance the solution to $t = 2\Delta t$, and so on

We do NOT know u so we can't simply integrate $\dot{u} = f(u,t) = -3u$ …

Activity: Why did we need to approximate $\dot{u} = f(u,t)$?

But we DO know $u(t_0) = u_0$ …

Activity: derive the Backward Euler method

The FE method approximates $\mathbf{f}$ as a constant using its value at $t = t_k$. Instead, BE treats it as a constant using its value at $t = t_{k+1}$.

In this setting:
(A) Derive the analog of equation (5)
(B) Develop a solution strategy, analogous to that of slide 5, for the IVP from equations (6) and (7)

$$\mathbf{u}_{k+1} = ??? \qquad (5)$$

Consider the IVP

$$\dot{u} = -3u$$

$$u(t = 0) = 4$$

Activity: derive the Backward Euler method

Equation (3) becomes

$$\mathbf{u}(t_{k+1}) - \mathbf{u}(t_k) = \mathbf{f}(\mathbf{u}(t_{k+1}), t_{k+1}) \int_{t_k}^{t_{k+1}} dt$$

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

This suggests a formula for finding the approximation to $\mathbf{u}(t_{k+1})$ given $\mathbf{u}(t_k)$:

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

And applying this to the IVP (6)-(7) gives

$$u_{k+1} = u_k + \Delta t(3u_{k+1})$$

$$\implies (1 + 3\Delta t)u_{k+1} = u_k$$

$$\implies u_{k+1} = \frac{u_k}{1 + 3\Delta t}$$

So we could start using $u_0 = u(t = 0) = 4$, and for some time step Δt could get $u_1, u_2, \ldots$

A note: ***explicit*** versus ***implicit*** methods

Let's compare the Forward and Backward Euler methods:

Evaluates $\mathbf{f}$ at the next time step t_{k+1}

Forward Euler	***Backward Euler***
$\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta t \mathbf{f}(\mathbf{u}_k, t_k)$	$\mathbf{u}_{k+1} = \mathbf{u}_k + \Delta t \mathbf{f}(\mathbf{u}_{k+1}, t_{k+1})$
For the IVP (6)-(7):	For the IVP (6)-(7):
$u_{k+1} = u_k - 3(\Delta t) u_k$	$u_{k+1} = u_k + \Delta t(3u_{k+1})$

Necessitates an extra step where u_{k+1} must be rearranged and solved for

Would be extra hard if $\mathbf{f}$ were nonlinear!
E.G., what if $f(u) = \log(u)^2$.
Would have to solve a nonlinear algebraic equation to get u_{k+1}

Because of the added complexity associated with methods that evaluate $\mathbf{f}$ at the next time step t_{k+1}, we categorize methods as follows:

Methods that ***do not*** involve the evaluation of $\mathbf{f}$ at $t = t_{k+1}$ are called ***explicit***.

Methods that ***do*** involve the evaluation of $\mathbf{f}$ at $t = t_{k+1}$ are called ***implicit***.

It is more costly to advance the system in time for implicit methods, but we will show later that there is a benefit in how ***stable*** the method is.

Another note: there are many more one-step methods!

Instead of interpolating $\mathbf{f}$ as a constant function, we could have represented it as a line or a higher order polynomial. These approaches lead to other methods!

Trapezoid method
(represent $\mathbf{f}$ as a line)

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{1}{2}\Delta t\Big(\boldsymbol{f}(\boldsymbol{u}_k, t_k) + \boldsymbol{f}(\boldsymbol{u}_{k+1}, t_{k+1})\Big)$$

4-stage Runge-Kutta method
(represent $\mathbf{f}$ as a higher degree polynomial)

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{1}{6}\Delta t\Big(\boldsymbol{y}_1 + 2\boldsymbol{y}_2 + 2\boldsymbol{y}_3 + \boldsymbol{y}_4\Big)$$

where

$$\begin{aligned}
\boldsymbol{y}_1 &= \boldsymbol{f}(\boldsymbol{u}_k, t_k) \\
\boldsymbol{y}_2 &= \boldsymbol{f}\left(\boldsymbol{u}_k + \frac{1}{2}\Delta t \boldsymbol{y}_1, t_k + \frac{1}{2}\Delta t\right) \\
\boldsymbol{y}_3 &= \boldsymbol{f}\left(\boldsymbol{u}_k + \frac{1}{2}\Delta t \boldsymbol{y}_2, t_k + \frac{1}{2}\Delta t\right) \\
\boldsymbol{y}_4 &= \boldsymbol{f}\left(\boldsymbol{u}_k + \Delta t \boldsymbol{y}_3, t_k + \Delta t\right)
\end{aligned}$$

See the typed notes for even more examples!