Today:

- *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):

\[
\frac{du}{dt} = f(u, t) \quad (1)
\]

\[
u(t_0) = u_0 \quad (2)
\]

We do NOT know \( u \)

We DO know \( f(u, t) \) and \( u(t_0) \)

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

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

Problem statement: finite-difference methods for IVPs

Given: \( u_k \approx u(t_k) = u(t_0 + k\Delta t) \)
Compute: \( u_{k+1} \approx u(t_{k+1}) = u(t_k + \Delta t) \).
How do we approximate \( u(t_{k+1}) \)?

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

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

\[
\Rightarrow u(t_{k+1}) - u(t_k) = \int_{t_k}^{t_{k+1}} f(u, t) dt \tag{3} \quad [\text{Fundamental theorem of calculus}]
\]

But what about the RHS? Interpolate \( f \)!

We will only consider local polynomial interpolation of \( f \) 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 \( f \).
The simplest one step finite difference method: Forward Euler

Let’s consider an example together.

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

$$f_a(u, t) = f(u(t_k), t_k)$$

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

Then (3) becomes

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

Could evaluate at other times as well!

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

$$u_{k+1} = u_k + \Delta t f(u_k, t_k)$$

(4)

(5)

Note. In (4) we use $u(t_k)$ whereas in (5) we use $u_k$.

The real $u$ at $t = t_k$

Our approximation to the real $u$ at $t = t_k$
An example involving the forward Euler method

Consider the IVP

\[ \dot{u} = -3u \quad (6) \]
\[ u(t = 0) = 4 \quad (7) \]

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 \( \Delta t \). Then our approximation to the solution at \( \Delta t \) is

\[ u_{\Delta t} = u_0 - 3u_0 \Delta t \]
\[ = 4 - 3(4)\Delta t \]

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

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

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

But we DO know \( u(t_0) = u_0 \ldots \)
Activity: derive the Backward Euler method

The FE method approximates $f$ 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)

$$u_{k+1} = ??? \quad (5)$$

Consider the IVP

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

$$u(t = 0) = 4$$
**Activity:** derive the Backward Euler method

Equation (3) becomes

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

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

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

\[ u_{k+1} = u_k + \Delta t f(u_{k+1}, t_{k+1}) \]

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 \( \Delta t \) could get \( u_1, u_2, \ldots \).
A note: \textit{explicit} versus \textit{implicit} methods

Let’s compare the Forward and Backward Euler methods:

\textbf{Forward Euler}
\[
\begin{align*}
  u_{k+1} &= u_k + \Delta t f(u_k, t_k) \\
  \text{For the IVP (6)-(7):} & \quad u_{k+1} = u_k - 3(\Delta t)u_k
\end{align*}
\]

\textbf{Backward Euler}
\[
\begin{align*}
  u_{k+1} &= u_k + \Delta t f(u_{k+1}, t_{k+1}) \\
  \text{For the IVP (6)-(7):} & \quad u_{k+1} = u_k + \Delta t(3u_{k+1})
\end{align*}
\]

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

Methods that \textbf{do not} involve the evaluation of \( f \) at \( t = t_{k+1} \) are called \textit{explicit}.

Methods that \textbf{do} involve the evaluation of \( f \) at \( t = t_{k+1} \) are called \textit{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 \textit{stable} the method is.
Another note: there are many more one-step methods!

Instead of interpolating \( f \) 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 \( f \) as a line)

\[
\begin{align*}
  u_{k+1} &= u_k + \frac{1}{2} \Delta t \left( f(u_k, t_k) + f(u_{k+1}, t_{k+1}) \right)
\end{align*}
\]

**4-stage Runge-Kutta method**
(represent \( f \) as a higher degree polynomial)

\[
\begin{align*}
  u_{k+1} &= u_k + \frac{1}{6} \Delta t \left( y_1 + 2y_2 + 2y_3 + y_4 \right)
\end{align*}
\]

where

\[
\begin{align*}
  y_1 &= f(u_k, t_k) \\
  y_2 &= f \left( u_k + \frac{1}{2} \Delta t y_1, t_k + \frac{1}{2} \Delta t \right) \\
  y_3 &= f \left( u_k + \frac{1}{2} \Delta t y_2, t_k + \frac{1}{2} \Delta t \right) \\
  y_4 &= f \left( u_k + \Delta t y_3, t_k + \Delta t \right)
\end{align*}
\]

See the typed notes for even more examples!