18. ODEs II: Improving on the Euler method

Summary of the previous lecture

- Ordinary differential equations: $\dot{x} = f(x)$
- The solution is a function $t \mapsto x(t)$

- Numerical methods approximate the solution
- Time stepping with step size h
- Euler method:

$$x_{n+1} = x_n + h\, f(x_n)$$

Goals for today

- Improving over Euler
- Trapezoid method
- Runge–Kutta methods

Euler method

- Recall: We want to solve

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t))$$

Euler method

- Recall: We want to solve

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t))$$

- By expanding to first order in a Taylor series we get the **Euler method**:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h_n \, \mathbf{f}(t_n, \mathbf{x}_{n+1})$$

Euler method

- Recall: We want to solve

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t))$$

- By expanding to first order in a Taylor series we get the **Euler method**:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h_n\, \mathbf{f}(t_n, \mathbf{x}_{n+1})$$

- The local truncation error at each step is $\mathcal{O}(h^2)$
- The global error is $\mathcal{O}(h)$

ODEs as integral equations

- Recall that we can rewrite the ODE $\dot{x} = f(x)$ as

$$x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} f(x(t))\,\mathrm{d}t$$

ODEs as integral equations

- Recall that we can rewrite the ODE $\dot{x} = f(x)$ as

$$x(t_{n+1}) = x(t_n) + \int_{t_n}^{t_{n+1}} f(x(t))\,\mathrm{d}t$$

- The Euler method corresponds to using the rectangular rule approximation

$$f(x(t)) \simeq f(x_n)$$

in $[x_n, x_{n+1}]$

Collaboration I

Approximating the integral

1. What is a better approximation for the integral?
2. What equation do we obtain if we do that?
3. Can we solve that?

Using the trapezoid rule

- We can approximate the integral in the interval $[x_n, x_{n+1}]$ using *any* quadrature method

Using the trapezoid rule

- We can approximate the integral in the interval $[x_n, x_{n+1}]$ using *any* quadrature method
- E.g. the trapezoid rule:

$$x_{n+1} = x_n + \frac{h}{2}\left[f(x_n) + f(x_{n+1})\right]$$

Using the trapezoid rule

- We can approximate the integral in the interval $[x_n, x_{n+1}]$ using *any* quadrature method
- E.g. the trapezoid rule:

$$x_{n+1} = x_n + \frac{h}{2}\left[f(x_n) + f(x_{n+1})\right]$$

- How can we find x_{n+1}? This is now an **implicit** equation for x_{n+1} in terms of itself!

Trapezoid rule II

- We *can* use this, but to do so we must *solve a nonlinear equation*!

Trapezoid rule II

- We *can* use this, but to do so we must *solve a nonlinear equation*!
- E.g. using the Newton method

Trapezoid rule II

- We *can* use this, but to do so we must *solve a nonlinear equation*!
- E.g. using the Newton method
- This is possible, but is expensive – the nonlinear equation must be solved at each step

Trapezoid rule II

- We *can* use this, but to do so we must *solve a nonlinear equation*!
- E.g. using the Newton method
- This is possible, but is expensive – the nonlinear equation must be solved at each step

- This turns out to be *necessary* for **stiff equations**: when there are multiple time scales

Trapezoid rule II

- We *can* use this, but to do so we must *solve a nonlinear equation*!
- E.g. using the Newton method
- This is possible, but is expensive – the nonlinear equation must be solved at each step

- This turns out to be *necessary* for **stiff equations**: when there are multiple time scales
- E.g. a pendulum hanging from a stiff spring

Trapezoid rule II

- We *can* use this, but to do so we must *solve a nonlinear equation*!
- E.g. using the Newton method
- This is possible, but is expensive – the nonlinear equation must be solved at each step

- This turns out to be *necessary* for **stiff equations**: when there are multiple time scales
- E.g. a pendulum hanging from a stiff spring
- The trapezoid rule has local error $\mathcal{O}(h^3)$ and global error $\mathcal{O}(h^2)$

Collaboration II

Making an explicit rule out of the trapezoid rule

1. Is there a way to make an explicit rule out of the trapezoid rule by using an additional approximation?

Modifying the trapezoid rule to make it explicit

- Let's look at the trapezoid rule again:

$$x_{n+1} = x_n + \frac{h}{2}\left[f(x_n) + f(x_{n+1})\right]$$

Modifying the trapezoid rule to make it explicit

- Let's look at the trapezoid rule again:

$$x_{n+1} = x_n + \frac{h}{2}\left[f(x_n) + f(x_{n+1})\right]$$

- This is implicit due to the term $f(x_{n+1})$ on the right-hand side
- Could we make this explicit by approximating it?

Modifying the trapezoid rule to make it explicit

- Let's look at the trapezoid rule again:

$$x_{n+1} = x_n + \frac{h}{2}\left[f(x_n) + f(x_{n+1})\right]$$

- This is implicit due to the term $f(x_{n+1})$ on the right-hand side
- Could we make this explicit by approximating it?
- We already have a way to approximate x_{n+1}, namely…

Modifying the trapezoid rule to make it explicit

- Let's look at the trapezoid rule again:

$$x_{n+1} = x_n + \frac{h}{2}\left[f(x_n) + f(x_{n+1})\right]$$

- This is implicit due to the term $f(x_{n+1})$ on the right-hand side
- Could we make this explicit by approximating it?

- We already have a way to approximate x_{n+1}, namely…
- … the Euler method! We can take an **Euler step**:

$$x_{n+1} \simeq x_n + h\, f(x_n)$$

The modified Euler method

- The idea of the **modified Euler** method is to use this approximation:

The modified Euler method

- The idea of the **modified Euler** method is to use this approximation:
- We first find an approximation for x_{n+1}, using one Euler step

The modified Euler method

- The idea of the **modified Euler** method is to use this approximation:
- We first find an approximation for x_{n+1}, using one Euler step
- Then we use that to get the next approximation for x_{n+1}

The modified Euler method

- The idea of the **modified Euler** method is to use this approximation:
- We first find an approximation for x_{n+1}, using one Euler step
- Then we use that to get the next approximation for x_{n+1}

- This gives a **multi-stage** method

Modified Euler II

- We use the notation k_i for successive evaluations of f at different points:

$$\begin{aligned} k_1 &:= f(x_n) \\ k_2 &:= f(x_n + h\,k_1) \end{aligned}$$

$$x_{n+1} = x_n + \tfrac{h}{2}(k_1 + k_2)$$

Modified Euler II

- We use the notation k_i for successive evaluations of f at different points:

$$\begin{aligned} k_1 &:= f(x_n) \\ k_2 &:= f(x_n + h\, k_1) \end{aligned}$$

$$x_{n+1} = x_n + \frac{h}{2}(k_1 + k_2)$$

- The k_i are evaluations of f
- Hence we are averaging approximations of $\dot{x}$ at different places

Modified Euler II

- We use the notation k_i for successive evaluations of f at different points:

$$k_1 := f(x_n)$$
$$k_2 := f(x_n + h\,k_1)$$

$$x_{n+1} = x_n + \frac{h}{2}(k_1 + k_2)$$

- The k_i are evaluations of f
- Hence we are averaging approximations of $\dot{x}$ at different places
- Note: Some references put h in the definition of k_i:

Modified Euler III

- When f has an explicit time dependence, $\dot{x}(t) = f(t, x(t))$, we must evaluate f at different ts too:

$$
\begin{aligned}
k_1 &:= f(t_n, x_n) \\
k_2 &:= f(t_n + h, x_n + h\, k_1)
\end{aligned}
$$

$$x_{n+1} = x_n + \frac{h}{2}(k_1 + k_2)$$

Collaboration III

Accuracy of modified Euler

1. How could we find how accurate modified Euler is as a function of h? Use the version without t_n.
2. What do we need to compare to?

Second-order Taylor methods

- The standard Euler method gives the Taylor expansion of $x(t)$ to *first* order

Second-order Taylor methods

- The standard Euler method gives the Taylor expansion of $x(t)$ to *first* order
- We hope that modified Euler reproduces the Taylor expansion of $x(t)$ to *second* order

Second-order Taylor methods

- The standard Euler method gives the Taylor expansion of $x(t)$ to *first* order
- We hope that modified Euler reproduces the Taylor expansion of $x(t)$ to *second* order
- Let's calculate that expansion for $\dot{x}(t) = f(t, x(t))$

Second-order Taylor methods

- The standard Euler method gives the Taylor expansion of $x(t)$ to *first* order
- We hope that modified Euler reproduces the Taylor expansion of $x(t)$ to *second* order
- Let's calculate that expansion for $\dot{x}(t) = f(t, x(t))$
- Expanding a single step to higher order we obtain

$$x(t+h) = x(t) + h\,\dot{x}(t) + \tfrac{1}{2}h^2\,\ddot{x}(t) + \mathcal{O}(h^3)$$

- How can we deal with $\ddot{x}(t)$?

Second-order Taylor methods II

- We know that $\dot{x}(t) = f(t, x(t))$

Second-order Taylor methods II

- We know that $\dot{x}(t) = f(t, x(t))$
- Differentiating the ODE we obtain

$$\ddot{x}(t) = \frac{d}{dt}\left[f(t, x(t))\right]$$

Second-order Taylor methods II

- We know that $\dot{x}(t) = f(t, x(t))$
- Differentiating the ODE we obtain

$$\ddot{x}(t) = \frac{d}{dt}\left[f(t, x(t))\right]$$

- We get $\ddot{x}(t) = f_t + f_x \dot{x}$

Second-order Taylor methods II

- We know that $\dot{x}(t) = f(t, x(t))$
- Differentiating the ODE we obtain

$$\ddot{x}(t) = \frac{d}{dt}\left[f(t, x(t))\right]$$

- We get $\quad \ddot{x}(t) = f_t + f_x \dot{x}$
- With the notation $f_t := \frac{\partial f}{\partial t}(t, x(t))$ and similarly for f_x

Second-order Taylor methods II

- We know that $\dot{x}(t) = f(t, x(t))$
- Differentiating the ODE we obtain

$$\ddot{x}(t) = \frac{d}{dt}\left[f(t, x(t))\right]$$

- We get $\quad \ddot{x}(t) = f_t + f_x \dot{x}$
- With the notation $f_t := \frac{\partial f}{\partial t}(t, x(t))$ and similarly for f_x
- So

$$x(t+h) \simeq x(t) + h\,f + \tfrac{1}{2}h^2\left[f_t + f_x\,f\right]$$

Second-order Taylor methods II

- We know that $\dot{x}(t) = f(t, x(t))$
- Differentiating the ODE we obtain

$$\ddot{x}(t) = \frac{d}{dt}\left[f(t, x(t))\right]$$

- We get $\quad \ddot{x}(t) = f_t + f_x \dot{x}$
- With the notation $f_t := \frac{\partial f}{\partial t}(t, x(t))$ and similarly for f_x
- So

$$x(t+h) \simeq x(t) + h\, f + \tfrac{1}{2} h^2 \left[f_t + f_x\, f\right]$$

- For clarity we write f instead of $f(t, x(t))$

Second-order Taylor methods II

- We know that $\dot{x}(t) = f(t, x(t))$
- Differentiating the ODE we obtain

$$\ddot{x}(t) = \frac{d}{dt}\left[f(t, x(t))\right]$$

- We get $\quad \ddot{x}(t) = f_t + f_x \dot{x}$
- With the notation $f_t := \frac{\partial f}{\partial t}(t, x(t))$ and similarly for f_x
- So

$$x(t+h) \simeq x(t) + h\, f + \tfrac{1}{2} h^2 \left[f_t + f_x\, f\right]$$

- For clarity we write f instead of $f(t, x(t))$

Order of modified Euler

- How accurate is the modified Euler method?

Order of modified Euler

- How accurate is the modified Euler method?
- Let's perform a Taylor expansion in h:

$$k_2 = f + h\, f_t + h\, k_1\, f_x + \mathcal{O}(h^2)$$

Order of modified Euler

- How accurate is the modified Euler method?
- Let's perform a Taylor expansion in h:

$$k_2 = f + h\, f_t + h\, k_1\, f_x + \mathcal{O}(h^2)$$

- So

$$\begin{aligned} x_{n+1} &= x_n + \tfrac{h}{2}(k_1 + k_2) \\ &\simeq x_n + \tfrac{h}{2}\left[f + f + h f_t + h k_1 f_x + \mathcal{O}(h^2)\right] \\ &= x_n + h\, f + \tfrac{1}{2} h^2 (f_t + f_x f) + \mathcal{O}(h^3) \end{aligned}$$

Order of modified Euler

- How accurate is the modified Euler method?
- Let's perform a Taylor expansion in h:

$$k_2 = f + h\, f_t + h\, k_1\, f_x + \mathcal{O}(h^2)$$

- So

$$\begin{aligned} x_{n+1} &= x_n + \tfrac{h}{2}(k_1 + k_2) \\ &\simeq x_n + \tfrac{h}{2}\left[f + f + hf_t + hk_1 f_x + \mathcal{O}(h^2)\right] \\ &= x_n + h\, f + \tfrac{1}{2}h^2(f_t + f_x f) + \mathcal{O}(h^3) \end{aligned}$$

- Hence modified Euler indeed *reproduces the second-order Taylor expansion*!

Order of modified Euler II

- Modified Euler reproduces the second-order Taylor expansion
- The local error is $\mathcal{O}(h^3)$ and the global error is $\mathcal{O}(h^2)$

Order of modified Euler II

- Modified Euler reproduces the second-order Taylor expansion
- The local error is $\mathcal{O}(h^3)$ and the global error is $\mathcal{O}(h^2)$
- We say that it is an **order-2** method

Order of modified Euler II

- Modified Euler reproduces the second-order Taylor expansion
- The local error is $\mathcal{O}(h^3)$ and the global error is $\mathcal{O}(h^2)$
- We say that it is an **order-2** method
- We only need evaluate f twice
- We never explicitly calculate derivatives!

General Runge–Kutta methods

- **Runge–Kutta** methods generalise this idea

General Runge–Kutta methods

- **Runge–Kutta** methods generalise this idea
- We will only discuss *explicit* Runge–Kutta methods

General Runge–Kutta methods

- **Runge–Kutta** methods generalise this idea
- We will only discuss *explicit* Runge–Kutta methods
- We want to reproduce the Taylor expansion of $x(t)$ up to some order

General Runge–Kutta methods

- **Runge–Kutta** methods generalise this idea
- We will only discuss *explicit* Runge–Kutta methods
- We want to reproduce the Taylor expansion of $x(t)$ up to some order
- By taking multiple Euler steps to calculate approximations of $\dot{x}(t)$

General Runge–Kutta methods

- **Runge–Kutta** methods generalise this idea
- We will only discuss *explicit* Runge–Kutta methods
- We want to reproduce the Taylor expansion of $x(t)$ up to some order
- By taking multiple Euler steps to calculate approximations of $\dot{x}(t)$
- And averaging them

General Runge–Kutta methods

- **Runge–Kutta** methods generalise this idea
- We will only discuss *explicit* Runge–Kutta methods
- We want to reproduce the Taylor expansion of $x(t)$ up to some order
- By taking multiple Euler steps to calculate approximations of $\dot{x}(t)$
- And averaging them
- The idea is to match the Taylor expansion

General Runge–Kutta methods

- **Runge–Kutta** methods generalise this idea
- We will only discuss *explicit* Runge–Kutta methods
- We want to reproduce the Taylor expansion of $x(t)$ up to some order
- By taking multiple Euler steps to calculate approximations of $\dot{x}(t)$
- And averaging them
- The idea is to match the Taylor expansion
- The algebra gets nasty!

Runge–Kutta methods II

- We can add more **stages** (evaluations of f)

Runge–Kutta methods II

- We can add more **stages** (evaluations of f)
- A general method with s stages is

$$
\begin{aligned}
k_1 &= f(t_n, x_n) \\
k_2 &= f(t_n + c_1 h, x_n + a_{11} k_1) \\
k_3 &= f(t_n + c_2 h, x_n + a_{21} k_1 + a_{22} k_2) \\
&\vdots \\
k_s &= f(t_n + c_1 h, x_n + a_{s-1,1} k_1 + \cdots + a_{s-1,s-1} k_{s-1}) \\
x_{n+1} &= x_n + h(b_1 k_1 + \cdots + b_s k_s)
\end{aligned}
$$

Runge–Kutta methods II

- We can add more **stages** (evaluations of f)
- A general method with s stages is

$$
\begin{aligned}
k_1 &= f(t_n, x_n) \\
k_2 &= f(t_n + c_1\, h, x_n + a_{11} k_1) \\
k_3 &= f(t_n + c_2\, h, x_n + a_{21} k_1 + a_{22} k_2) \\
&\vdots \\
k_s &= f(t_n + c_1\, h, x_n + a_{s-1,1} k_1 + \cdots + a_{s-1,s-1} k_{s-1}) \\
x_{n+1} &= x_n + h(b_1 k_1 + \cdots + b_s k_s)
\end{aligned}
$$

- The a_{ij}, b_i and c_i satisfy certain constraints

Butcher tableau

- The coefficients can be laid out in a **Butcher tableau**:

$$
\begin{array}{c|ccccc}
0 & & & & \\
c_1 & a_{11} & & & \\
c_2 & a_{21} & a_{22} & & \\
\vdots & & & & \\
c_{s-1} & a_{s-1,1} & \cdots & a_{s-1,s-1} & \\
\hline
 & b_1 & b_2 & b_{s-1} & b_s
\end{array}
$$

Butcher tableau

- The coefficients can be laid out in a **Butcher tableau**:

$$
\begin{array}{c|ccccc}
0 & & & & & \\
c_1 & a_{11} & & & & \\
c_2 & a_{21} & a_{22} & & & \\
\vdots & & & & & \\
c_{s-1} & a_{s-1,1} & \cdots & a_{s-1,s-1} & & \\
\hline
 & b_1 & b_2 & b_{s-1} & b_s &
\end{array}
$$

- E.g. for modified Euler we get

$$
\begin{array}{c|cc}
0 & & \\
1 & 1 & \\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}
$$

4th-order Runge–Kutta method: RK4

- Matching Taylor expansions for higher-order methods requires tedious algebra
- There are clever techniques to do so

4th-order Runge–Kutta method: RK4

- Matching Taylor expansions for higher-order methods requires tedious algebra
- There are clever techniques to do so
- An elegant and efficient 4th-order method that is commonly used is RK4:

$$\begin{aligned} k_1 &= f(t_n, x_n) \\ k_2 &= f(t_n + h/2, x_n + k_1/2) \\ k_3 &= f(t_n + h/2, x_n + k_2/2) \\ k_4 &= f(t_n + h, x_n + k_3) \end{aligned}$$

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

RK4 II

- This is simpler to understand and implement as a Butcher tableau:

$$
\begin{array}{c|cccc}
0 & & & & \\
\frac{1}{2} & \frac{1}{2} & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & \\
1 & 0 & 0 & 1 & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

Summary

- Trapezoid method: Has a smaller error and is more stable than Euler, but is **implicit**
- Approximating the trapezoid method gives a 2nd-order method that is explicit

- Runge–Kutta methods: take nested Euler steps
- We can reproduce Taylor expansions to different orders