ICT DATE CHANGE

Note that the ICT date changed, it is now Wednesday 13/5/2026, from 1 to 4 pm, in INB2101.

You should use the university-provided computers and not your own laptop, so make sure you are confortable using them.

RECAP/FEEDBACK LAST SESSION

EXERCISE 1 LAST SESSION

For Euler, being a first order integration algorithm:, the error is proportional to Δt, $\epsilon = c_1 \Delta t$ (with c_1 some constant). Hence halving the timestep halves the error, so the error is 0.02.

Ralston is a second order method, so $\epsilon = c_2 \Delta t^2$ (with c_2 some constant). Now halving the timestep would reduce the error by a factor of 4, so the error is $0.03/4 = 0.0075$.

GENERAL HINT FOR LAST SESSION

If you didn't manage to reach agreement with the analytical solution for the last session for sufficiently small time step, do the following exercise.

Solve the ODE by pen-and-paper, using only *one* integration step (so from $t = 0$ to $t = \Delta t = 0.01$),

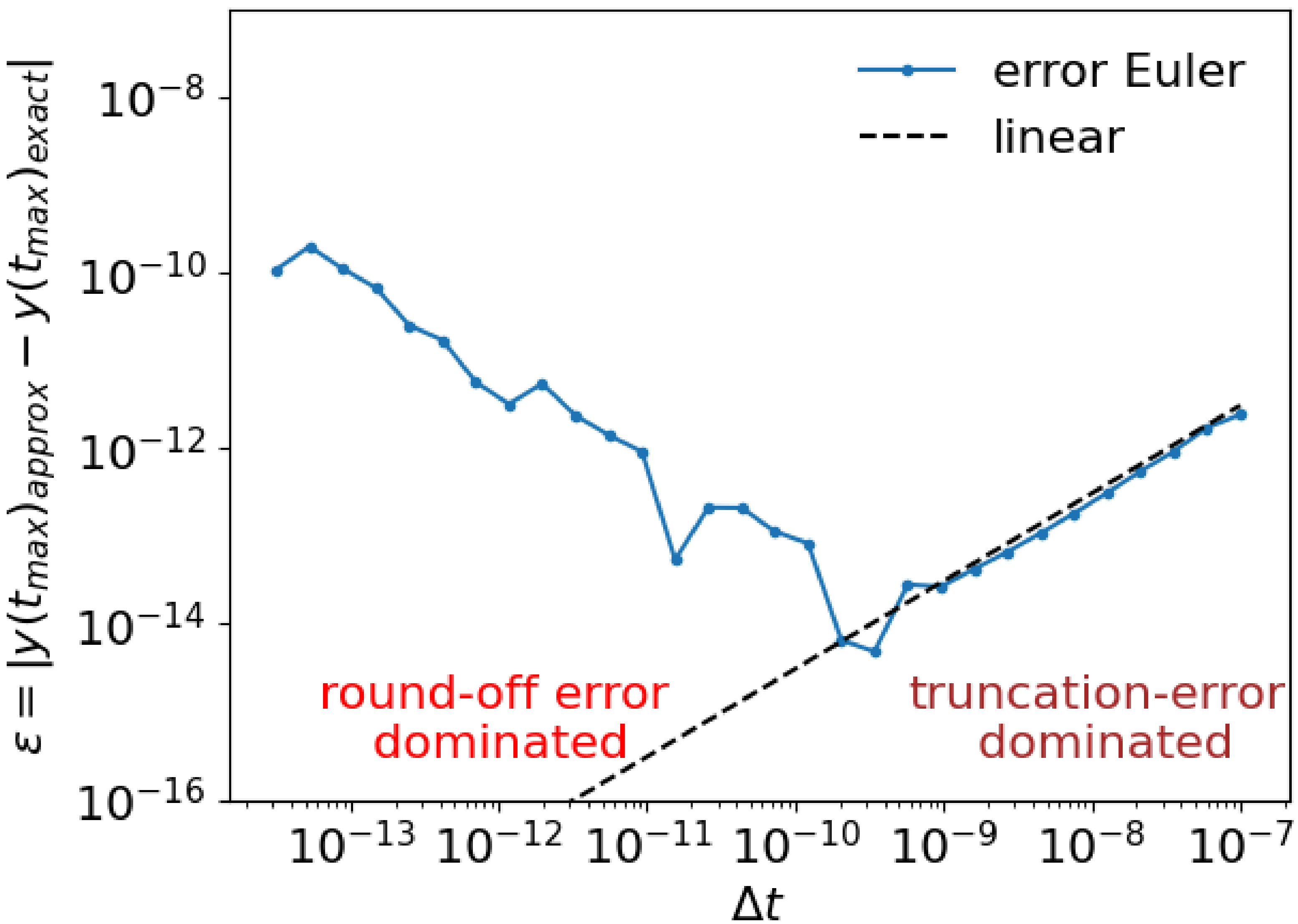
- explicit Euler
- Ralston

Compare it with the computer program that you made, and ensure they give the same results. You can even compare the values of k_1 and k_2, if necessary. You may have to compare two steps, so till $t = 2\Delta t$. This is a good way to ensure that your program gives the desired result.

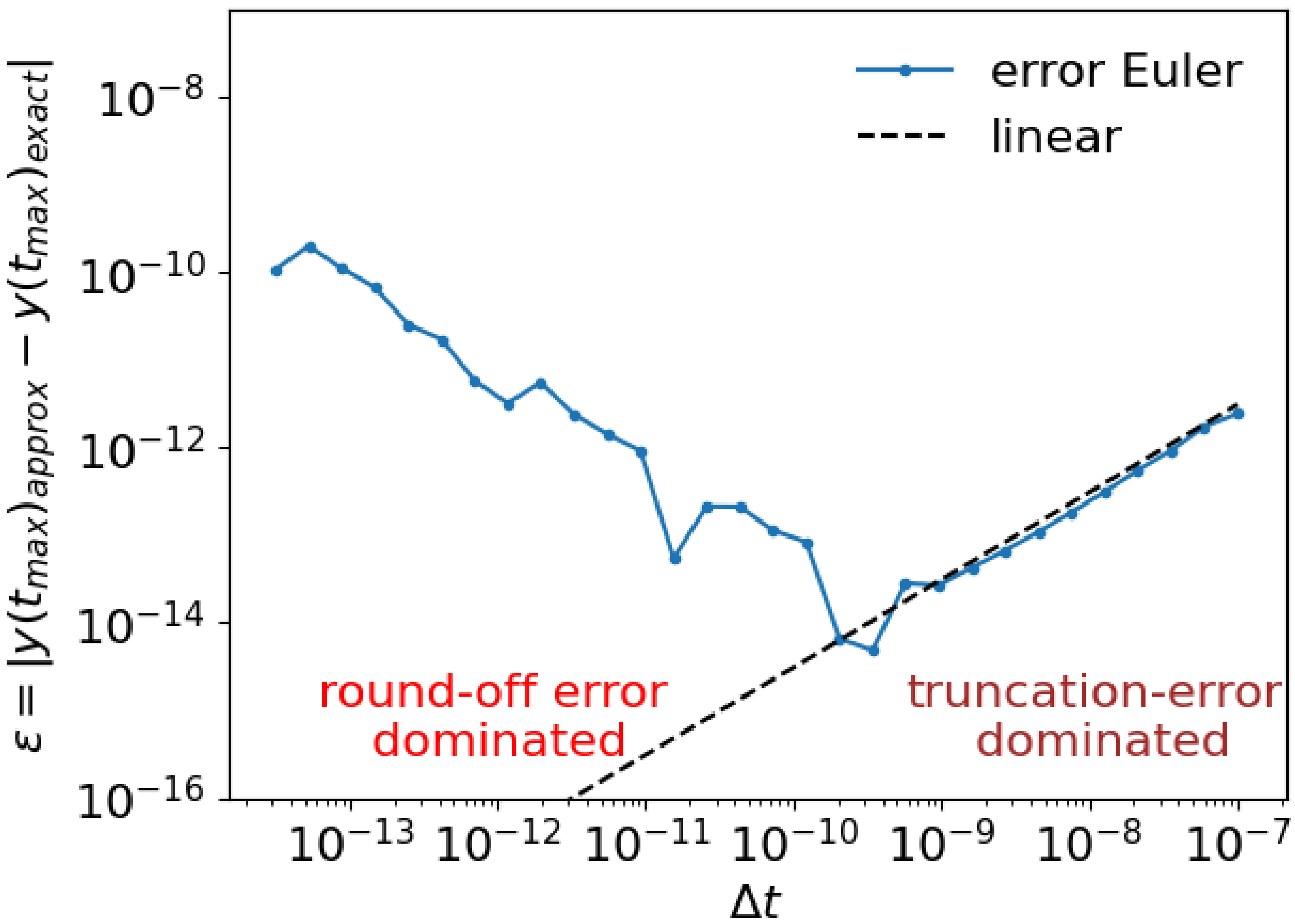
ROUND-OFF ERRORS

The **round-off errors** arise because of *a finite representation of a floating point number* in a programming language. Round-off errors do not only occur for rounding numbers to their integer value, but also due to rounding a number to a limited fractional value. An example is rounding π to 3.1416.

ROUND-OFF ERROR ILLUSTRATED BY THE EXPLICIT EULER METHOD



error Euler
linear
round-off error dominated
truncation-error dominated
$\varepsilon = |y(t_{max})_{approx} - y(t_{max})_{exact}|$
Δt
10^{-8}
10^{-10}
10^{-12}
10^{-14}
10^{-16}
10^{-13} 10^{-12} 10^{-11} 10^{-10} 10^{-9} 10^{-8} 10^{-7}



From $\Delta t \lesssim 10^{-9}$ the error does not decrease any more with smaller time-step. The reason: limited accuracy of the 64 bit floating point number is reached. For $\Delta t \approx 10^{-10}$ the error in the calculation $\epsilon \approx 10^{-13}$.

The error is much larger than the threshold for a `np.float64`, $\approx 10^{-16}$. This is caused by the *accumulation* of round-off errors due to the many additions of order N_{int}. Since N_{int} increases with smaller Δt, the *error will even increase* with smaller Δt.

However, this is only the case for very small Δt and in this module the main focus will be on truncation errors.

HIGHER ORDER METHODS

HIGHER ORDER METHODS

RUNGE KUTTA METHODS

Runge-Kutta methods (which are explicit) can be cast in the form

$$\boxed{y_{i+1} \approx y_i + \Delta t \cdot \phi(t_i, y_i, \Delta t)} \qquad (*)$$

where ϕ is called the *increment function*

MIDPOINT METHOD

The **second order Runge Kutta midpoint method** for the ODE $\dot{y}(t) = g(t, y(t))$ is:

$$\boxed{y_{i+1} \approx y_i + \Delta t \cdot g(t_i + \Delta t/2, y_i + g(t_i, y_i)\Delta t/2)}$$

Now the function g is evaluated at around half the time step in between t_i and t_{i+1}, where this half point is approximated by the derivative at t_i. So $\phi(t_i, y_i, \Delta t) = g(t_i + \Delta t/2, y_i + g(t_i, y_i)\Delta t/2)$.

RALSTON'S METHOD

Instead of halfway, we can evaluate it at 2/3th of the interval. One can show that then the local truncation error has a minimum bound, and the resulting method is called **Ralston's method**:

$$\boxed{y_{i+1} \approx y_i + \Delta t \cdot \left(\frac{1}{4} g(t_i, y_i) + \frac{3}{4} g(t_i + \frac{2}{3}\Delta t, y_i + \frac{2}{3} g(t_i, y_i)\Delta t)\right)}$$

This can be written as

$$y_{i+1} \approx y_i + \Delta t \cdot \left(\frac{1}{4} k_1 + \frac{3}{4} k_2\right)$$

with

$$\begin{aligned} k_1 &= g(t_i, y_i) \\ k_2 &= g(t_i + \frac{2}{3}\Delta t, y_i + \frac{2}{3}\Delta t \cdot k_1) \end{aligned}$$

TODAY

- Derivation 2nd order RK methods
- 4th order RK method
- Symmetric method

DERIVATION 2ND ORDER RK METHODS

The above equations can be derived as follows

GENERAL FORM

For the second order RK methods the increment function equals

$$\phi(y_i, t_i, \Delta t) = a_1 k_1(t_i, y_i) + a_2 k_2(t_i, y_i, \Delta t) \tag{1}$$

where the a's are constants, and the k's are

$$\begin{aligned} k_1 &= g(t_i, y_i) \\ k_2 &= g(t_i + p_1 \Delta t, y_i + q_{11} \Delta t \cdot k_1) \end{aligned}$$

such that

$$\begin{aligned} y_{i+1} &\approx y_i + \Delta t \cdot (a_1 k_1 + a_2 k_2) \\ &\approx y_i + \Delta t \cdot a_1 g(t_i, y_i) + \Delta t \cdot a_2 g(t_i + p_1 \Delta t, y_i + q_{11} \Delta t \cdot k_1) \end{aligned}$$

DERIVATION OF COEFFICIENTS

We need to determine the unknown coefficients a_1, a_2, p_1, and q_{11}

TAYLOR EXPANSION ODE

For the ODE $y'(t) = g(t, y(t))$ we can apply a Taylor series to determine $y_{i+1} \equiv y(t_{i+1})$ from $y_i \equiv y(t_i)$

$$\begin{aligned} y_{i+1} &= y_i + \Delta t y_i' + \Delta t^2 y_i''/2! + O(\Delta t^3) \\ &= y_i + \Delta t g_i + \Delta t^2 g_i'/2! + O(\Delta t^3) \qquad \text{using } y' = g \end{aligned}$$

Here the time-derivative g' has to be determined by the *total* derivative, since y also changes when t changes:

$$\begin{aligned} g'(t_i, y_i) \equiv \frac{dg(t_i, y_i)}{dt_i} &= \frac{\partial g(t_i, y_i)}{\partial t_i} + \frac{\partial g(t_i, y_i)}{\partial y_i}\frac{\partial y_i}{\partial t_i} \\ &= \frac{\partial g(t_i, y_i)}{\partial t_i} + \frac{\partial g(t_i, y_i)}{\partial y_i} g(t_i, y_i) \end{aligned}$$

Hence the Taylor expansion becomes

$$y_{i+1} = y_i + \Delta t g_i + \frac{\Delta t^2}{2!}\left(\frac{\partial g(t_i, y_i)}{\partial t_i} + \frac{\partial g(t_i, y_i)}{\partial y_i} g(t_i, y_i)\right) + O(\Delta t^3)$$

TAYLOR EXPANSION METHOD

We have the RK2 scheme

$$y_{i+1} \approx y_i + \Delta t \cdot a_1 g(t_i, y_i) + \Delta t \cdot a_2 k_2 \qquad (*)$$

with $k_2 = g(t_i + p_1 \Delta t, y_i + q_{11} \Delta t \cdot k_1)$ and with $k_1 = g(t_i, y_i)$.

The function k_2 is of the form $f(x_1 + \Delta x_1, x_2 + \Delta x_2)$ and can be expanded as a Taylor series as well (in both variables)

$$k_2 \equiv g(t_i + p_1 \Delta t, y_i + q_{11} \Delta t k_1) = g(t_i, y_i) + p_1 \Delta t \frac{\partial g(t_i, y_i)}{\partial t_i} + q_{11} \Delta t k_1 \frac{\partial g(t_i, y_i)}{\partial y_i} + O(\Delta t^2)$$

Substituting this expansion for k_2 in the RK2 scheme (*) gives

$$y_{i+1} \approx y_i + \Delta t \cdot (a_1 + a_2) g(t_i, y_i) + \Delta t^2 \left(a_2 p_1 \frac{\partial g(t_i, y_i)}{\partial t_i} + a_2 q_{11} g(t_i, y_i) \frac{\partial g(t_i, y_i)}{\partial y_i} \right) + O(\Delta t^3)$$

EQUATING THE TAYLOR EXPANSIONS

The first few terms of the two Taylor expansion should then be set equal to each other. So equating the ODE

$$y_{i+1} = y_i + \Delta t g_i + \Delta t^2 \left(\frac{1}{2} \frac{\partial g(t_i, y_i)}{\partial t_i} + \frac{1}{2} \frac{\partial g(t_i, y_i)}{\partial y_i} g(t_i, y_i) \right) + O(\Delta t^3)$$

with the method

$$y_{i+1} \approx y_i + \Delta t \cdot (a_1 + a_2) g(t_i, y_i) + \Delta t^2 \left(a_2 p_1 \frac{\partial g(t_i, y_i)}{\partial t_i} + a_2 q_{11} g(t_i, y_i) \frac{\partial g(t_i, y_i)}{\partial y_i} \right) + O(\Delta t^3)$$

gives

$$\begin{aligned} a_1 + a_2 &= 1 \\ a_2 p_1 &= \frac{1}{2} \\ a_2 q_{11} &= \frac{1}{2} \end{aligned}$$

These are four unknowns and three equations: one parameter free (say a_2): $a_1 = 1 - a_2$, $p_1 = q_{11} = \frac{1}{2a_2}$. Hence there are infinitely many 2nd order RK methods.

COEFFICIENTS FOR AFOREMENTIONED METHODS

Midpoint method correspond to $a_2 = 1$ and hence $a_1 = 0$.

Ralston method correspond to $a_2 = 3/4$ and hence $a_1 = 1/4$.

ANY OTHER CONSIDERATIONS?

We looked at Runge Kutta methods, and they can be generalized to any order.

RK4

The **RK4** method for the ODE $y(t)' = g(t, y(t))$ is

$$y_{i+1} \approx y_i + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where

$$\begin{aligned} k_1 &= g(t_i, y_i) \\ k_2 &= g(t_i + \Delta t/2, y_i + \Delta t k_1/2) \\ k_3 &= g(t_i + \Delta t/2, y_i + \Delta t k_2/2) \\ k_4 &= g(t_i + \Delta t, y_i + \Delta t k_3) \end{aligned}$$

This is a fourth order method (global truncation error $O(\Delta t^4)$), and widely used.

SYMMETRIC METHODS

A method of the form

$$y_{i+1} \approx \Phi(\Delta t, y_i)$$

is called a **symmetric method** or **time-reversible** if exchanging $i+1 \leftrightarrow i$ and $\Delta t \leftrightarrow -\Delta t$ leaves the method invariant. Then integrating one step forward and subsequently one step backward would give the same result as the starting point.

WHY IMPORTANT?

Many differential equations are reversible in time (if replacing t by $-t$, then the same differential equation is obtained) and hence the numerical method will be obeying a similar long-time behaviour.

Example DE: all differential equations of the form $u(t)'' = g(u(t))$ are reversible. E.g., Newton's equation of motion for which u is the position and the derivative is with respect to time.

WHAT FOR THE RK FAMILY?

One can show that *all* of the explicit Runge Kutta methods are *not* symmetric. Hence they all can suffer from long-time drifts.

IMPLICIT TRAPEZOID METHOD

The implicit **trapezoid method** for the ODE $\dot{y}(t) = g(t, y(t))$ is

$$\boxed{y_{i+1} \approx y_i + \frac{\Delta t}{2}\big(g(t_i, y_i) + g(t_{i+1}, y_{i+1})\big)} \qquad (*)$$

Notice that this method is *implicit* (as with the implicit Euler method), since y_{i+1} also occurs at the RHS. Therefore, the equation $(*)$ has to be made *explicit* in y_{i+1} by isolating it to the LHS before one can actually implement it (analogous to the implicit Euler).

The method is second order, global truncation error $O(\Delta t^2)$.

PROOF THAT TRAPEZOID IS SYMMETRIC

The benefit of the trapezoid method is that y_i and y_{i+1} play a symmetric role; there is no bias towards either of them. For example, one can rewrite

$$y_{i+1} \approx y_i + \frac{\Delta t}{2}\big(g(t_i, y_i) + g(t_{i+1}, y_{i+1})\big)$$

as

$$y_i \approx y_{i+1} - \frac{\Delta t}{2}\big(g(t_i, y_i) + g(t_{i+1}, y_{i+1})\big)$$

by bringing both y_{i+1} and y_i to the other side and muliplying the result by -1. Note that this is exactly the same equation, but now i and $i+1$ have swapped and Δt is replaced by $-\Delta t$, since this corresponds to integrating backward in t. Hence it is a *symmetric* method.

This in contrast to the aforementioned explicit methods (such as the family of RK methods, which have a bias towards y_i) and the implicit Euler method (bias towards y_{i+1}): all these methods lack this symmetry.

PROOF THAT EXPLICIT EULER IS NOT SYMMETRIC

Explicit Euler is

$$y_{i+1} \approx y_i + \Delta t g(t_i, y_i)$$

Now this equation can be rewritten as (move y_{i+1} to the other side, same for y_i, and multiply both sides by -1)

$$y_i \approx y_{i+1} - \Delta t g(t_i, y_i) \tag{*}$$

Now integrating backwards using this algorithm would imply y_i needs to be known a priori to determine g and hence this is actually the implicit Euler variant and not the explicit Euler variant any more. Therefore it is not symmetric under time reversal.

Another way of seeing this is that the explicit Euler variant for integrating backwards would be (swap i and $i+1$ and replace Δt by $-\Delta t$)

$$y_i \approx y_{i+1} - \Delta t g(t_{i+1}, y_{i+1}) \tag{2}$$

and this equation is obviously not equal to eq. *.

EXERCISES SESSION 3

See Blackboard, assessment section.