Single-step methods for ODE integration.

Serguei Ossokine (Albert-Einstein-Institute).
Prepared for the Computational Physics IMPRS Course (2022).

We look in more detail at convergence of one-step explicit methods when solving ODEs. In PS1, you have already seen the Forward Euler, Midpoint and RK4 methods.

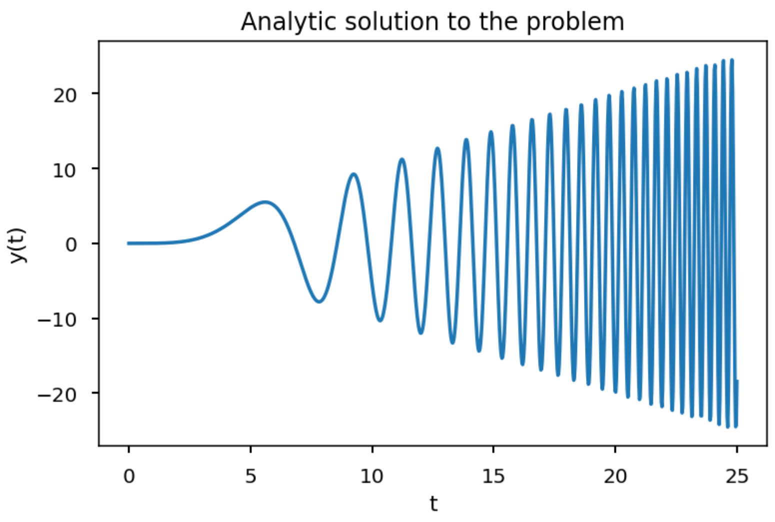
We now introduce an IVP where the amplitude and frequency of the solution grow with time, mimicking a chirping gravitational wave:

```
def F(t, y,info):
    """ This function returns the RHS of the differential equation
    dy/dt = func(t, y), for which the solution is y = exact_sol(t) below.
    """
    return 3/100 * t**3 * np.cos(1/100 * t**3) + np.sin(1/100 * t**3)
def exact_sol(t):
    """ Exact solution for a test case, chosen because that it has
    varying amplitude and time scales. In particular the ampltude grows
    proportionally to t, while the frequency grows proportionally to t**2.
    """
    Amp = t                         # Amplitude
    omega = 1/100 * t**2            # Frequency (rad/s)
    return Amp * np.sin(omega * t)
```

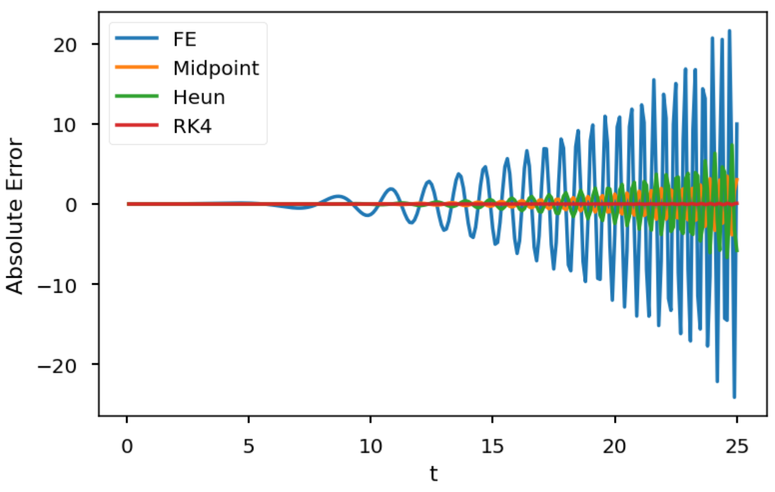
We will solve this problem with the following times and intial conditions:

```
t0 = 0                    # Start time
tmax = 25                 # End time
y0 = exact_sol(t0)        # initial value
```

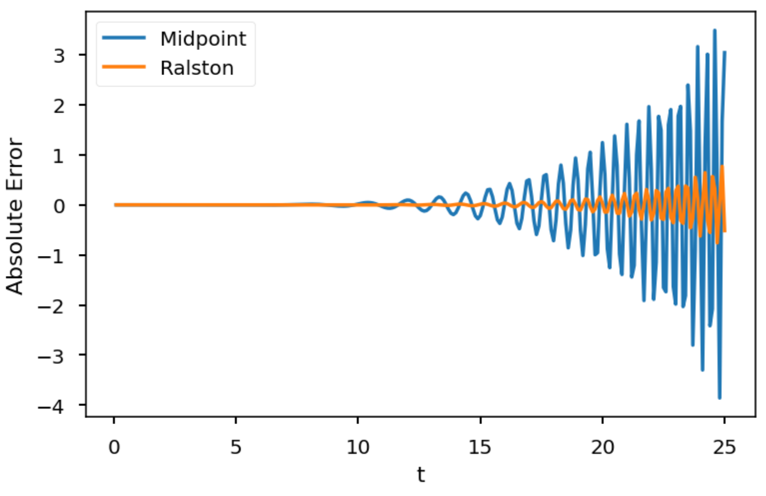
Here is the plot of the analytic solution.



Let's solve this problem numerically with different steppers from the lectures.



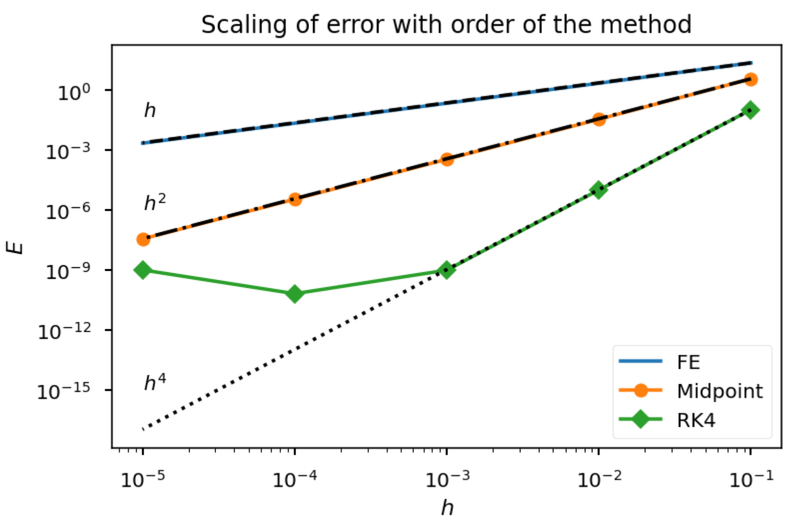
Unusurprisingly the higher order method performed better, achieving a much lower error. One important thing to keep in mind is that at a given order the RK methods are *not* unique and some may be suited better for a given problem. Indeed, let's try with a very particular 2nd order RK method, Ralston's method.



As we can see, the error is significantly better. It turns out that Ralston's method minimizes the *local* truncation error at 2nd order as discussed for example in Section 3 of the original paper.

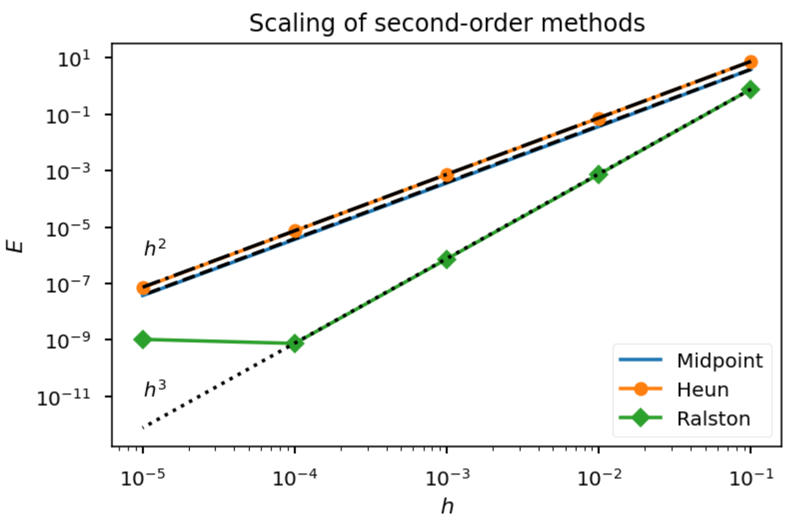
(Side note: there is some inconsistency on what people call the "Ralston method". Here we have followed the original paper definition)

Let's now check that we get the global truncation error scaling we expect from these methods.



We do indeed see the expected scaling, except for RK4 for $h \lesssim \times 10^{-3}$. Can you think of why the error might start growing?

Something interesting happens when we also compare our second-order steppers:



Notice that for Ralston's method, the error seems to scale as $\mathcal{O}(h^3)$ even though the method is 2nd order. Why did this happen?

The reason for this behaviour is that the particular choice of the stages of this RK2 method has reduced the constant in front of the $\mathcal{O}(h^3)$ term in the *local truncation error* so much that the higher order term dominates for this problem.

Note that this is not guaranteed for all problems.