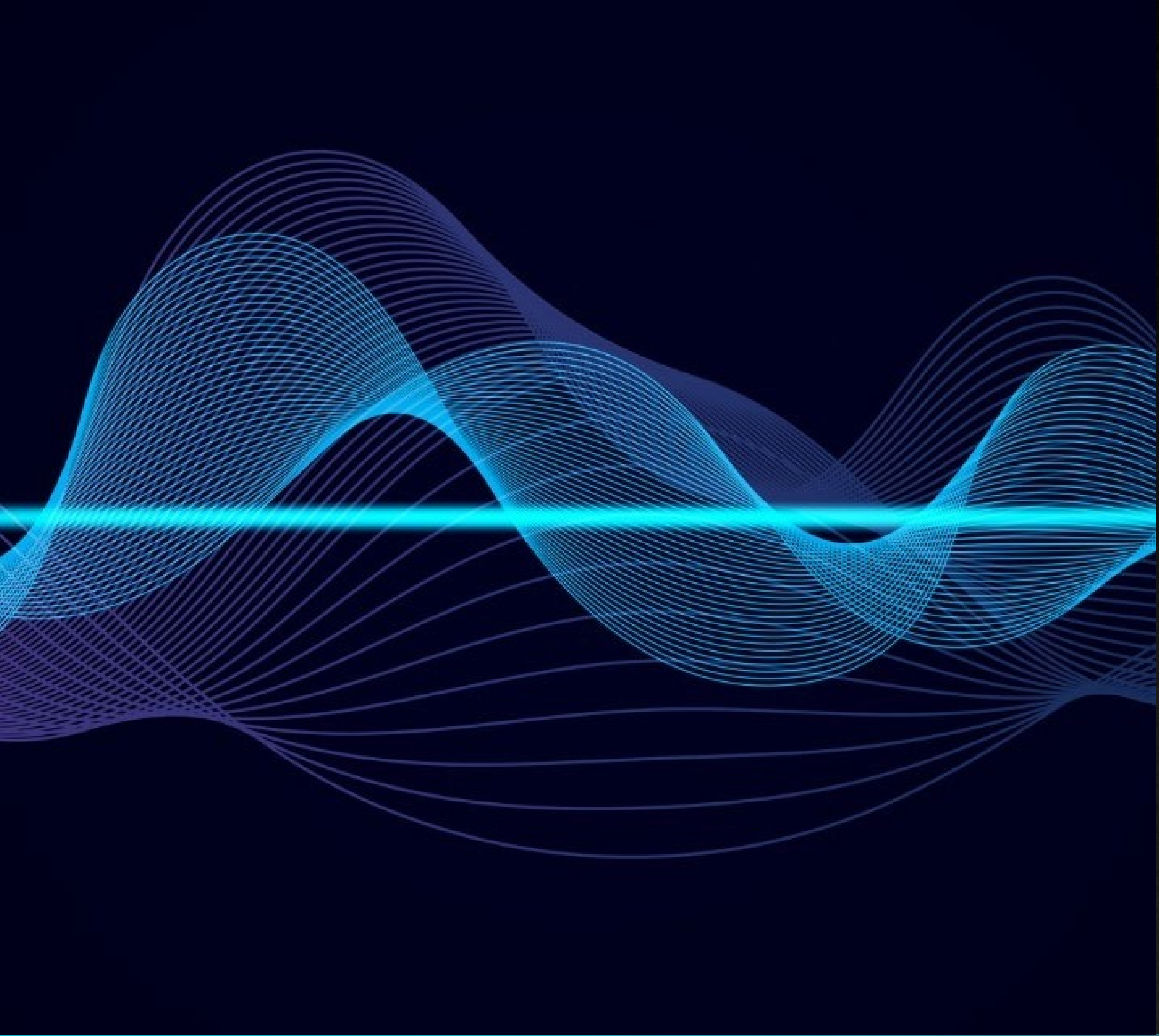


Computational Physics

PHYS 6260

Solving ODEs

Announcements:

- No class: 1/14-16 (travel), 1/20 (MLK)
- HW 2: Due Friday 1/24

Lecture Outline

We will cover these topics

- 1st order ODEs with 1 variable
  - Euler's method
  - Runge-Kutta method
- ODEs with multiple variables
- Higher-order ODEs
- Variable step sizes
- Leapfrog method (energy conserving)

1^{st} order ODEs

- Perhaps the most common task for computational physics is the solution of differential equations
- There are many methods, and we will cover only a few examples in this lecture

- An ordinary differential equation (ODE) has only one independent variable, such as time
  - It may contain other variables that are dependent on this independent variable, though

- The simplest type of ODE is a 1^{st} order equation with one dependent variable, such as

$$\frac{dx}{dt} = \frac{2x}{t}$$

- This can be easily solved analytically by separating the variables

1st order ODEs

- But it is common for ODEs that aren't separable, such as

$$\frac{dx}{dt} = \frac{2x}{t} + \frac{3x^2}{t^3}$$

- This is also non-linear.
- We can solve it numerically. First we need the ODE in the form dx/dt = f(x, t)
- For now, we will focus on time-independent solutions.
- To compute a solution, we need a set of initial condtions (analytical or numerical)

1^{st} order ODEs

Euler's Method

- The most straightforward method is Euler's method
- Evolves x with its derivate evaluated at time t with some timestep h

- We write the Taylor expansion around time t to calculate the next value of x at time t+h

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{1}{2}h^2\frac{d^2x}{dt^2} + \ \ldots$$

$$x(t+h) = x(t) + h\frac{dx}{dt} + O(h^2)$$

- For Euler's method, we neglect all terms higher than h^2

$$x(t+h) = x(t) + h\frac{dx}{dt}$$

In-class problem

Euler's Method

$$x(t+h) = x(t) + h\frac{dx}{dt}$$

- Use Euler's method to solve the ODE:

$$\frac{dx}{dt} = -x^3 + \sin t$$

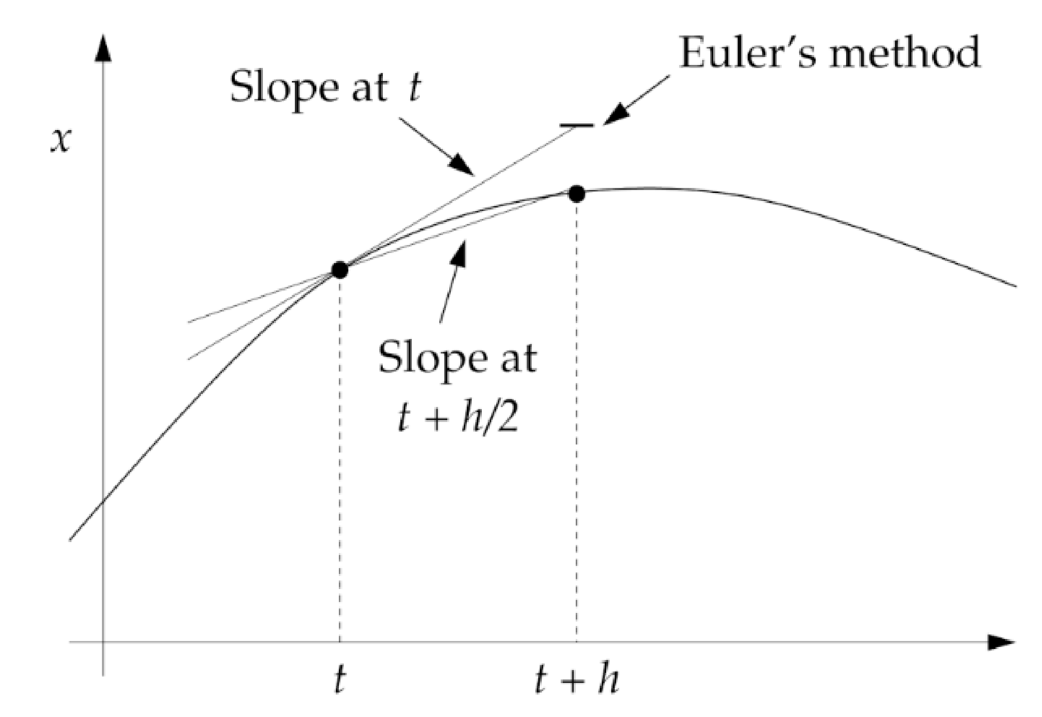
- Consider the initial condtion of x = 0 at t = 0
- Numerically integrate the system from t = 0 → 10 with 1000 steps
- Start with the skeleton code `04_euler0.py` on Canvas

- Running this program gives a good approximation to the actual solution
- In general, Euler's method is not a bad one, and in many cases, it is quite accurate
- However, it's not widely used because the higher-order Runge-Kutta method is easily implemented

1^{st} order ODEs
Runge-Kutta method

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{1}{2}h^2\frac{d^2x}{dt^2}$$

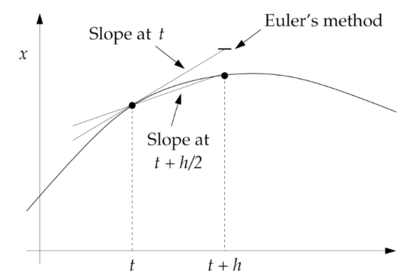
- We can improve Euler's method by keeping the 2^{nd} order term
- We can use df/dt instead of d^2x/dt^2
- The Runge-Kutta (RK) method is a general method
  - Technically Euler's method is the 1^{st} order RK method
  - The 2^{nd} order RK method is also known as the midpoint method
- As illustrated to the right, we can estimate the next x-value with the derivative at the midpoint



1^{st} order ODEs

Runge-Kutta method

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{1}{2}h^2\frac{d^2x}{dt^2}$$



- We can estimate the value of x(t+h) by taking the Taylor expansion of both x(t) and x(t+h) around the midpoint t + h/2

$$x(t+h) = x(t+\tfrac{1}{2}h) + \tfrac{1}{2}h\left(\frac{dx}{dt}\right)_{t+h/2} + \tfrac{1}{8}h^2\left(\frac{d^2x}{dt^2}\right)_{t+h/2} + O(h^3)$$

$$x(t) = x(t+\tfrac{1}{2}h) - \tfrac{1}{2}h\left(\frac{dx}{dt}\right)_{t+h/2} + \tfrac{1}{8}h^2\left(\frac{d^2x}{dt^2}\right)_{t+h/2} + O(h^3)$$

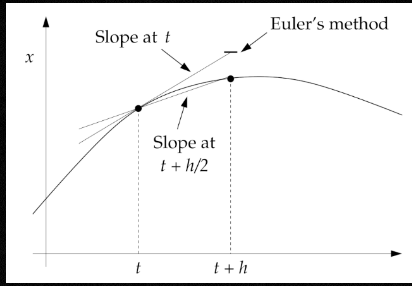
- Subtracting the 2^{nd} expression from the first, we arrive at

$$\begin{aligned} x(t+h) &= x(t) + h\left(\frac{dx}{dt}\right)_{t+h/2} + O(h^3) \\ &= x(t) + hf[x(t+\tfrac{1}{2}h), t+\tfrac{1}{2}h] + O(h^3) \end{aligned}$$

1^{st} order ODEs

Runge-Kutta method

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{1}{2}h^2\frac{d^2x}{dt^2}$$



- The h^2 terms have vanished so that the method has $O(h^3)$ errors now

$$\begin{aligned} x(t+h) &= x(t) + h\left(\frac{dx}{dt}\right)_{t+h/2} + O(h^3) \\ &= x(t) + hf[x(t+\tfrac{1}{2}h), t+\tfrac{1}{2}h] + O(h^3) \end{aligned}$$

- How do we calculate the slope at the midpoint?
  - We can calculate it with Euler's method!

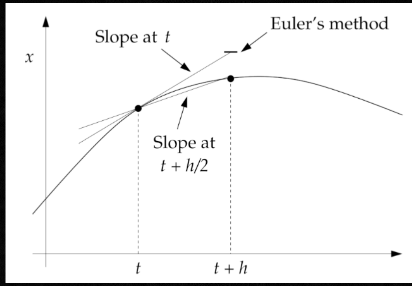
$$x\left(t+\frac{h}{2}\right) = x(t) + \frac{1}{2}hf(x,t)$$

  - We then use it in the equation above

1^{st} order ODEs

Runge-Kutta method

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{1}{2}h^2\frac{d^2x}{dt^2}$$



- This is the 2^{nd} order RK method

$$\begin{aligned} k_1 &= hf(x,t), \\ k_2 &= hf(x + \tfrac{1}{2}k_1, t + \tfrac{1}{2}h), \\ x(t+h) &= x(t) + k_2. \end{aligned}$$

- It's called 2^{nd} order because it's accurate to h^2

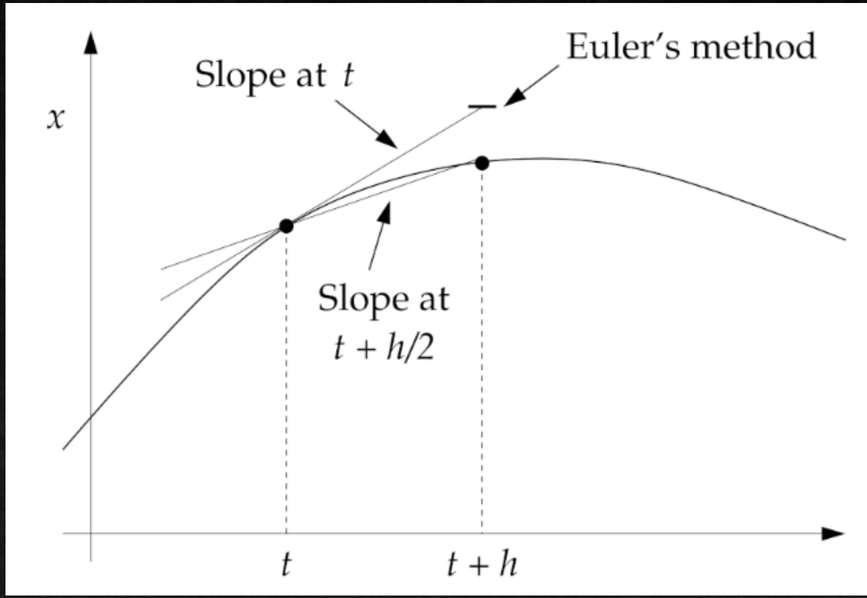
1^{st} order ODEs

4^{th} order Runge-Kutta method

- We can take even higher-order terms in the Taylor expansion
- The "sweet spot" is 4^{th} order that gives very high accuracy without much complexity
- It is the most widely used method to solve ODEs
- There are five equations to solve for the next timestep

$$\begin{aligned}
k_1 &= hf(x,t),\\
k_2 &= hf(x+\tfrac{1}{2}k_1, t+\tfrac{1}{2}h),\\
k_3 &= hf(x+\tfrac{1}{2}k_2, t+\tfrac{1}{2}h),\\
k_4 &= hf(x+k_3, t+h),\\
x(t+h) &= x(t) + \tfrac{1}{6}(k_1+2k_2+2k_3+k_4).
\end{aligned}$$

- This gives an accurate solution for the example for only N = 20 steps!



Multi-variable ODEs

- Many physics problems have multiple dependent variables -- a system of ODEs. For example,

$$\frac{dx}{dt} = xy - x, \qquad \frac{dy}{dt} = y - xy + \sin^2(\omega t)$$

- There is only one independent variable, t

- A general form for two 1st order ODEs is

$$\frac{dx}{dt} = f_x(x, y, t), \qquad \frac{dy}{dt} = f_y(x, y, t)$$

- We can further generalize this into an arbitrary number of dependent variables by putting the variables and functions into vectors:

$$\vec{r} = (x, y, \dots), \qquad \vec{f}(\vec{r}, t) = (f_x(\vec{r}, t), f_y(\vec{r}, t), \dots)$$

Multi-variable ODEs

- Put the variables and functions into vectors:

$$\vec{r} = (x, y, \ldots), \qquad \vec{f}(\vec{r}, t) = (f_x(\vec{r}, t),\ f_y(\vec{r}, t), \ldots)$$

- Thus we can compactly express the system as

$$\frac{d\vec{r}}{dt} = \vec{f}(\vec{r}, t)$$

- We can use this form on any RK method. For example, the 4th order method reads as

$$\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f}(\mathbf{r}, t), \\
\mathbf{k}_2 &= h\mathbf{f}(\mathbf{r} + \tfrac{1}{2}\mathbf{k}_1, t + \tfrac{1}{2}h), \\
\mathbf{k}_3 &= h\mathbf{f}(\mathbf{r} + \tfrac{1}{2}\mathbf{k}_2, t + \tfrac{1}{2}h), \\
\mathbf{k}_4 &= h\mathbf{f}(\mathbf{r} + \mathbf{k}_3, t + h), \\
\mathbf{r}(t + h) &= \mathbf{r}(t) + \tfrac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4).
\end{aligned}$$

In-class example

System of ODEs with RK4

- Use the skeleton code `04_RK4.py` on Canvas to solve the system of ODEs

$$\frac{dx}{dt} = xy - x, \qquad \frac{dy}{dt} = y - xy + \sin^2(\omega t)$$

- Use numpy's vector notation
- Use the initial condition of x = 1, y = 1 at t = 0
- Use a frequency of $\omega = 1$
- Integrate from t = 0 → 10

$$\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f}(\mathbf{r}, t), \\
\mathbf{k}_2 &= h\mathbf{f}(\mathbf{r} + \tfrac{1}{2}\mathbf{k}_1, t + \tfrac{1}{2}h), \\
\mathbf{k}_3 &= h\mathbf{f}(\mathbf{r} + \tfrac{1}{2}\mathbf{k}_2, t + \tfrac{1}{2}h), \\
\mathbf{k}_4 &= h\mathbf{f}(\mathbf{r} + \mathbf{k}_3, t + h), \\
\mathbf{r}(t+h) &= \mathbf{r}(t) + \tfrac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4).
\end{aligned}$$

Solving 2nd order ODEs

- So far, we've focused on 1st order ODEs, but these are rare in physics
- 2nd order and higher ODEs are more common
- Solving these are an extension of the 1st order methods

- Consider a 2nd order ODE with one independent variable t

$$\frac{d^2x}{dt^2} = f\left(x, \frac{dx}{dt}, t\right)$$

- Meaning that the 2nd derivative can be any arbitrary function, including non-linear ones

Solving 2nd order ODEs

- For example, consider

$$\frac{d^2x}{dt^2} = \frac{1}{x}\left(\frac{dx}{dt}\right)^2 + 2\frac{dx}{dt} - x^2e^{-4t}$$

- We can put this equation in the form $\frac{d^2x}{dt^2} = f\left(x, \frac{dx}{dt}, t\right)$ by defining $y \equiv \frac{dx}{dt}$ that leads to

$$\frac{dy}{dt} = f(x, y, t)$$

- That is *exactly the same* as a 1st order ODE

Solving higher-order ODEs

- We can use a similar approach for 3rd and higher order ODEs

$$\frac{d^3x}{dt^3} = f\left(x, \frac{dx}{dt}, \frac{d^2x}{dt^2}, t\right)$$

- This will have two additional independent variables that are the 1st and 2nd derivatives
  - $y \equiv \frac{dx}{dt}, z \equiv \frac{dy}{dt}$
- Leading to a system of three 1st order ODEs

$$\frac{dz}{dt} = f(x, y, z, t)$$

Solving higher-order ODEs

- We can generalize this to a vector form where we can consider an arbitrary number of dependent variables and higher order derivatives

$$\frac{d^2\vec{r}}{dt^2} = \vec{f}\left(\vec{r}, \frac{d\vec{r}}{dt}, t\right)$$

- This is equivalent to the 1st order equations

$$\frac{d\vec{r}}{dt} = \vec{s}, \qquad \frac{d\vec{s}}{dt} = \vec{f}(\vec{r}, \vec{s}, t)$$

- Given a system of n equations of mth order, we would have a set of m x n simulataneous 1st order equations that we can use conventional ODE and matrix solvers

In-class problem
The non-linear pendulum

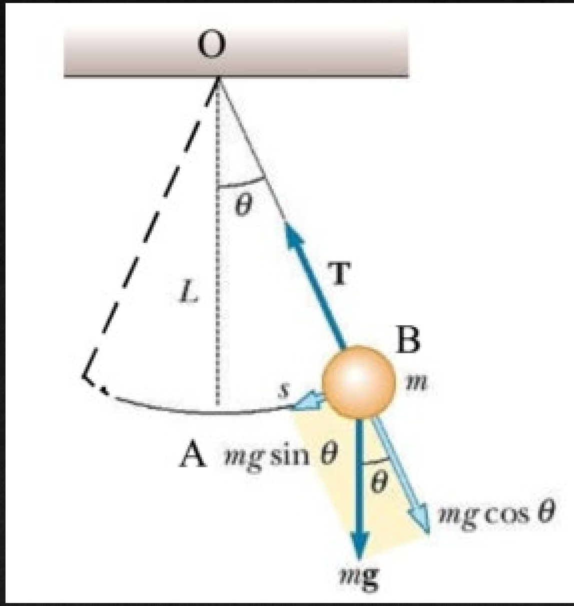
- A standard physics problem is the linear pendulum
  - Approximates the behavior with a linear ODE that can be solved exactly

- The equation of motion is

$$ml\frac{d^2\theta}{dt^2} = -mg\,\sin\theta$$
$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\,\sin\theta$$

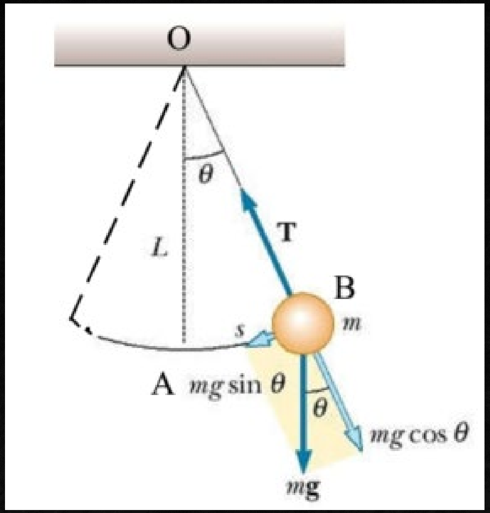
- We can re-write this as two 1st order ODEs

$$\frac{d\theta}{dt} = \omega, \qquad \frac{d\omega}{dt} = -\frac{g}{l}\,\sin\theta$$



In-class problem

The non-linear pendulum



- $\frac{d\theta}{dt} = \omega, \quad \frac{d\omega}{dt} = -\frac{g}{l}\sin\theta$

- We can combine these two variables into a single vector $\vec{r} = (\theta, \omega)$
- Use the RK4 method to solve these two equations simulataneously

- We are only interested in θ though
- Consider the case where the arm is 10 cm
- Initial condition: $\theta = 179°, \omega = 0$
- Use the skeleton code `04_pendulum0.py`

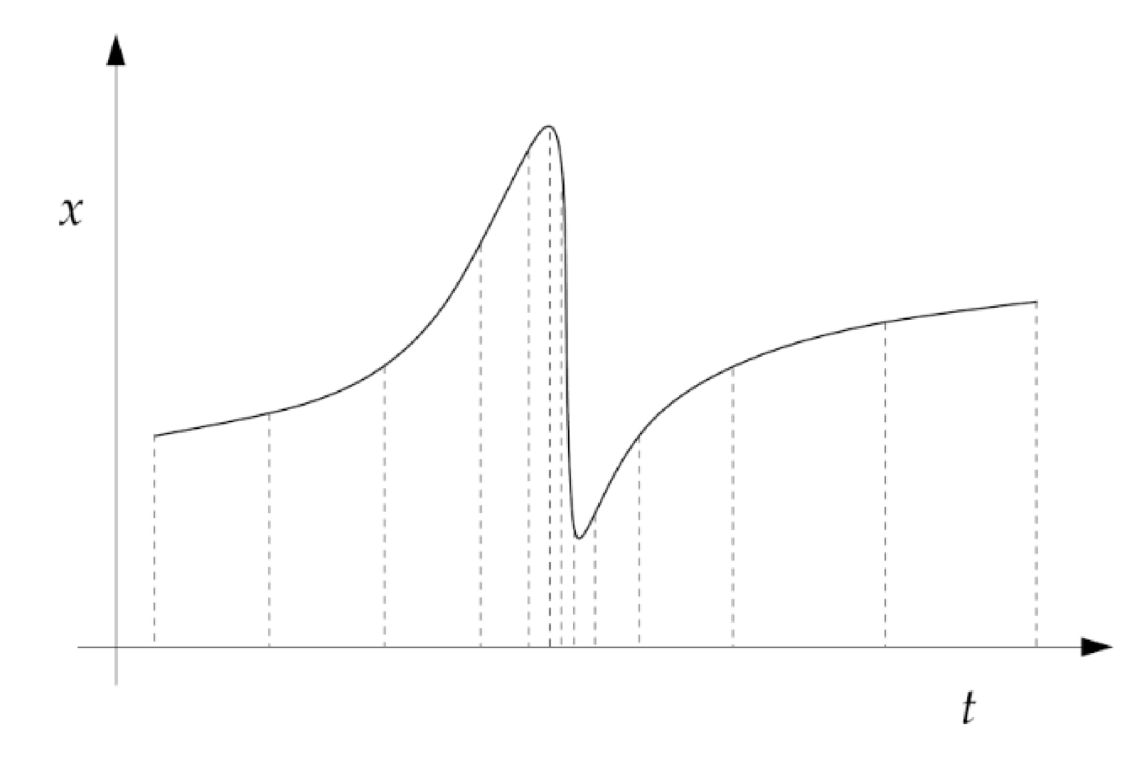
I'm a Comet!

Varying the step size

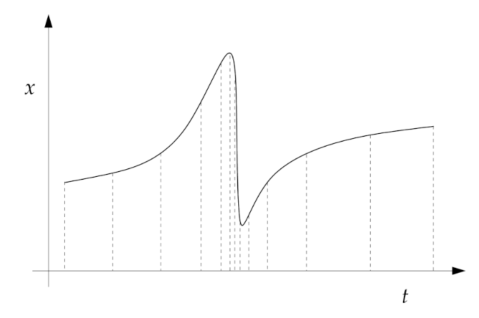
- So far, we've been using methods with constant step sizes
- We can achieve much higher accuracy if we allow the step size to vary

- Suppose that we're solving a 1st order ODE: $\frac{dx}{dt} = f(x,t)$ shown to the right

- When the slope is steep, a smaller stepsize will prevent overshooting the actual solution
- Otherwise we can take longer steps and maintain accuracy while being efficient



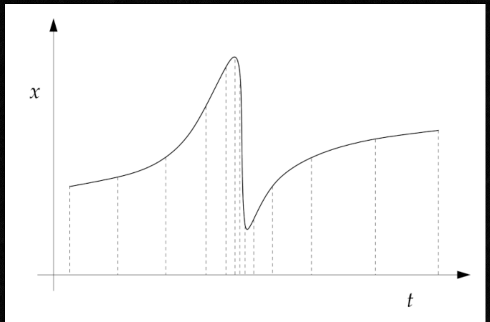
Varying the step size



- The idea behind an adaptive step size is to keep a constant error per unit interval t
- In practice, there are two parts to this
  - Estimate the error and compare to our desired accuracy
  - Increase/decrease the step size to maintain this accuracy

- Let's look at the RK4 method as an example
  - Choose some initial step h
  - Take two steps to time t + 2h
  - Go back to time t and take a single step 2h
  - Compare the results of x(t + 2h)

Varying the step size



- RK4 is a 4th order method that has 5th order errors: ch^5 for a single timestep, where c is an unknown constant.

- Therefore, for a two timesteps of size h, we have the numerical solution x_1 and the error

$$x(t+2h) = x_1 + 2ch^5$$

- For a single timestep of 2h,

$$x(t+2h) = x_2 + 32ch^5$$

- Equating these two solutions, we find that the error per timestep is

$$\epsilon \equiv ch^5 = \frac{1}{30}(x_1 - x_2)$$

- Our goal is to adjust h so that ϵ is closer but never greater than a target accuracy

Varying the step size

- Consider a target accuracy δ per unit time
- Let's call the "perfect" timestep h' that achieves this accuracy

$$\epsilon' = ch'^5 = ch^5\left(\frac{h'}{h}\right)^5 = \frac{1}{30}(x_1 - x_2)\left(\frac{h'}{h}\right)^5$$

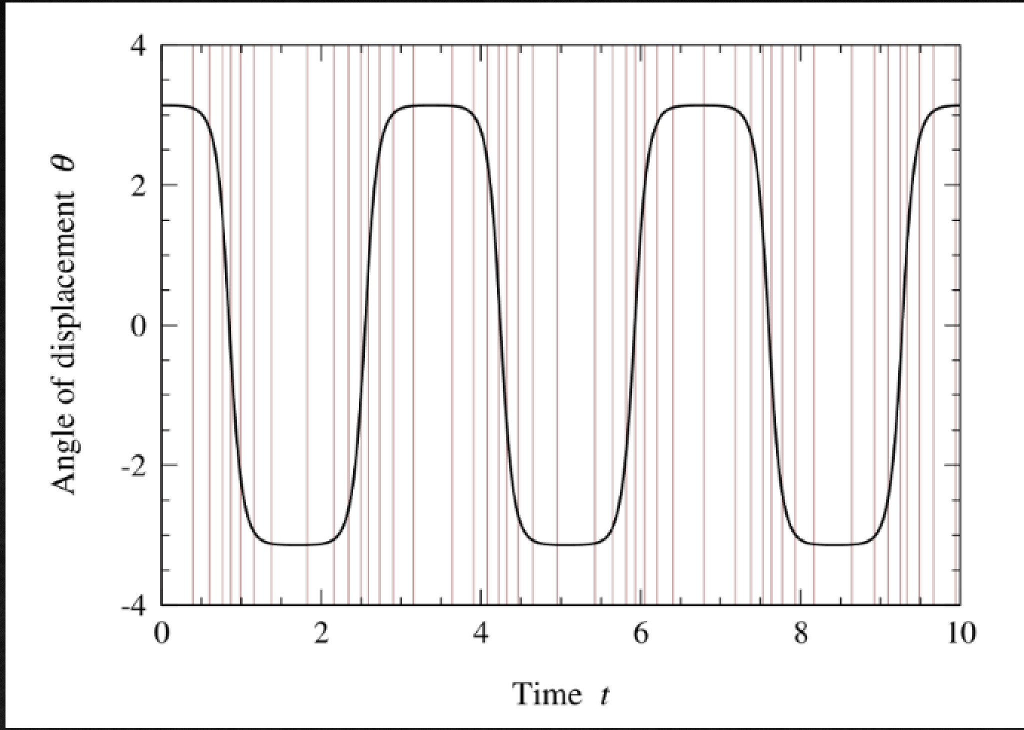
- Set this error equal to the target accuracy

$$\frac{1}{30}|x_1 - x_2|\left(\frac{h'}{h}\right)^5 = h'\delta$$

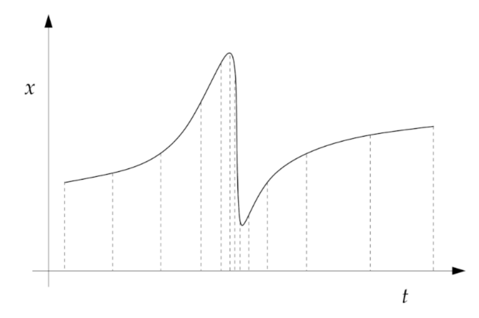
- Solve for h'

$$h' = h\left(\frac{30h\delta}{|x_1 - x_2|}\right)^{1/4} = h\rho^{1/4}$$

- Where we've defined $\rho \equiv 30h\delta/|x_1 - x_2|$



Varying the step size
Summary



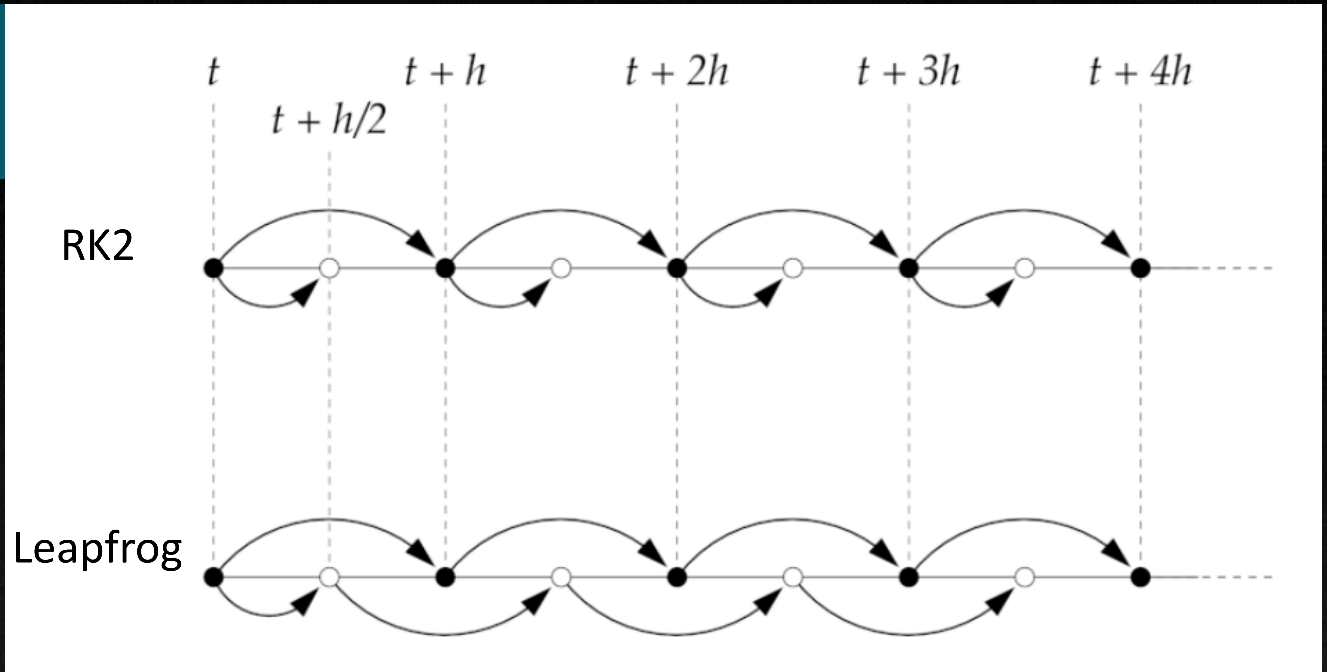
1. Perform two steps of size h and one step of size 2h, starting at the same point
2. From these calculations, compute ρ from the two estimates x_1 and x_2
3. If $\rho > 1$, the actual accuracy is better than the target one
   1. Keep the x_1 estimate (2 steps)
   2. To avoid computational waste in the next step, we increase h by the factor ρ
4. If $\rho < 1$, the actual accuracy is poor than the target one. We have to repeat the calculation with a smaller timestep, decreased by the factor ρ
5. Note: usually the step size is not allowed to change more than a factor of two

Error accumulation

- The RK method gives a straightforward and robust way to integrate ODEs
- But in the pendulum example, we can show that it doesn't necessarily conserve energy
- The methods covered today conserve energy
  - Ideal for long integration periods

- Consider the 1st order ODE: $dx/dt = f(x,t)$
- Solving it with RK2 that requires dx/dt at the midpoint
  - 2nd order accurate method
  - We estimate the midpoint slope every step
  - The associated errors accumulate
  - Total error is only 1st order accurate

Leapfrog method

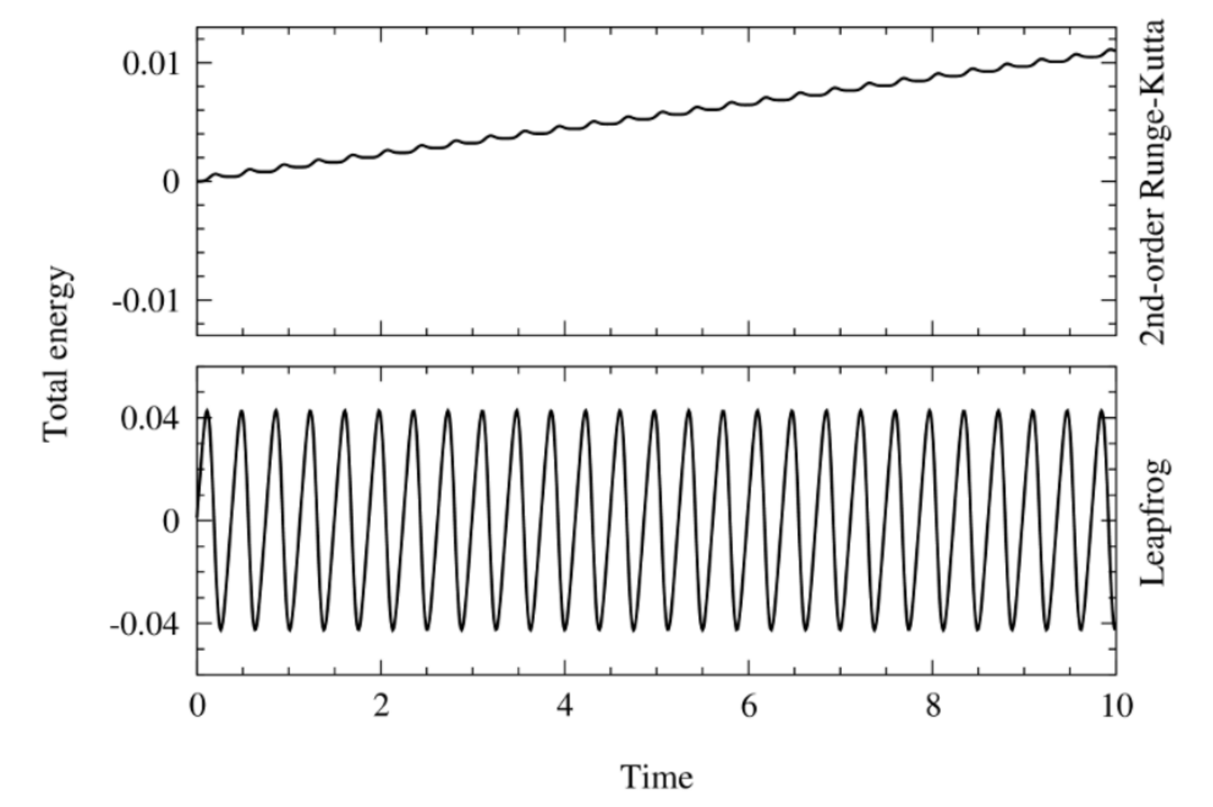


- We can avoid accumulating errors by not taking the half timesteps
- Have two simulataneous solutions
  - At integer steps
  - At half-integer steps
- Then we have estimates of the slopes at the midpoints for both solutions
- Results in the total error remaining 2nd order accurate
- Conserves energy

$$x(t + \tfrac{1}{2}h) = x(t) + \tfrac{1}{2}f(x,t)$$
$$x(t + h) = x(t) + hf(x(t + \tfrac{1}{2}h), t + \tfrac{1}{2}h).$$
$$x(t + \tfrac{3}{2}h) = x(t + \tfrac{1}{2}h) + hf(x(t + h), t + h).$$
$$x(t + 2h) = x(t + h) + hf(x(t + \tfrac{3}{2}h), t + \tfrac{3}{2}h).$$

Leapfrog method

- The leapfrog method is time-reversal symmetric → conserves energy
  - One can show this by going forward in time +h with the governing equations and then backwards with –h to recover the original equations
  - Runge-Kutta methods do not have this property
- In general, the leapfrog method should be used for any periodic system
- Provides a stable long-term solution, conserving energy



Verlet method

- Suppose that we're using the leapfrog method to solve a 2nd order ODE, like Newton's second law

$$\frac{d^2x}{dt^2} = f(x,t)$$

- That can be written as two coupled 1st order ODEs

$$\frac{dx}{dt} = v, \qquad \frac{dv}{dt} = f(x,t)$$

- By defining the vector $\vec{r} = (r, v)$, we write it as a single expression

$$\frac{d\vec{r}}{dt} = \vec{f}(\vec{r},t)$$

Verlet method

- Let's inspect the numerics of the system (leapfrog method)
- The position at the next timestep is given by

$$x(t+h) = x(t) + hv\left(t + \frac{1}{2}h\right)$$

- To advance to (t + 2h) we need the velocity at the next midpoint

$$v\left(t + \frac{3}{2}h\right) = v\left(t + \frac{1}{2}h\right) + hf(x(t+h), t+h)$$

- Notice that we never need the velocity at integer timesteps
- This is a special case where (1) the RHS of the 1st equation only depends on v not x and (2) the RHS of the 2nd equation only depends on x not v
- *Many physics problems have this form*

Verlet method

- The only downside to this method happens if we need a quantity that depends on v at full timesteps, like energy

- One can estimate v with Euler's method from a half-timestep

$$v(t+h) = v\left(t+\frac{1}{2}h\right) + \frac{1}{2}hf(x(t+h), t+h)$$

- In summary
  - Calculate the velocity at the first half-timestep: $v\left(t+\frac{h}{2}\right) = v(t) + \frac{1}{2}hf(x(t),t)$
  - Then the subsequent values of x and v are calculated with

$$\begin{aligned} x(t+h) &= x(t) + hv(t+\tfrac{1}{2}h) \\ k &= hf(x(t+h), t+h) \\ v(t+h) &= v(t+\tfrac{1}{2}h) + \tfrac{1}{2}k \\ v(t+\tfrac{3}{2}h) &= v(t+\tfrac{1}{2}h) + k \end{aligned}$$

Modified midpoint method

- Because the leapfrog method has the nice property of time reversalibility, its error is

$$\epsilon(-h) = -\epsilon(h)$$

- Telling us that the error is an odd function

$$\epsilon(h) = c_3 h^3 + c_5 h^5 + c_7 h^7 + \ \ldots$$

- where c_i are constants

- There is one catch where we need to use Euler's method to estimate the slope at the first midpoint.
  - Introduces errors on the order of h^2
- We can cancel these errors out in the last timestep (see lecture notes / book for details)
- Not a popular method because it few benefits over leapfrog and is less accurate than RK4