Numerical Methods for Astrophysics:

ORDINARY DIFFERENTIAL EQUATIONS (ODEs)
Part 1

Michela Mapelli
Ordinary Differential Equations (ODEs). Concept

ODEs ARE UBIQUITOUS IN ASTROPHYSICS

Examples:

– equation of motion of a particle subject to Newton’s gravitational force

\[
\frac{d^2 x_i}{dt^2} = -G \sum_{j=1, j\neq i}^{N} m_j \frac{x_i - x_j}{|x_i - x_j|^3}
\]

– equation of hydrostatic equilibrium of a star interior

\[
\frac{dP}{dr} = -G \frac{M \rho}{r^2}
\]
ODEs. Euler Method

General form of an ODE in 1 variable: \[ \frac{dx}{dt} = f(x, t) \]

For simplicity, let’s assume \( t \) is time

Simplest way to proceed: TAYLOR EXPANSION

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

or with different notation

\[ x(t + h) = x(t) + h f(x, t) + \mathcal{O}(h^2) \]

If we neglect higher order terms we can calculate \( x(t+h) \) as

\[ x(t + h) = x(t) + h f(x, t) \]

equation of the Euler scheme

if we know the value of \( x \) at time \( t \), then we can derive the value of \( x \) at time \( (t+h) \)

This approximation is “good” if \( h \) is small
ODEs. Euler Method

Equation of the Euler’s method: \( x(t + h) = x(t) + h \, f(x, t) \)

To integrate the ODE between \( t = t_0 \) and \( t = t_f \)

I need to choose a \( h << (t_f - t_0) \)

and then to repeat Euler’s equation for \( N \) steps with

\[ N = \frac{(t_f - t_0)}{h} \]

Euler equation is 1\(^{\text{st}}\) order method → errors scale as \( h^2 \)
→ we can reduce the error by reducing \( h \)
   but if we reduce \( h \) the computing time increases

We will see other methods that have a smaller error
   for the same (or similar) computing time
ODEs. Runge-Kutta family

Family of algorithms that solve ODEs via Taylor’s expansion

**First order:** Euler method or first-order Runge-Kutta method

**Second order:** midpoint or second-order Runge-Kutta method

**Fourth order:** fourth-order Runge-Kutta method

etcetc
ODEs. Midpoint or second-order Runge-Kutta method

Evaluate the slope $dx/dt$ of $x(t)$ not at the end of the interval $h$, but at the midpoint of the interval $h/2$

Mathematically, corresponds to a Taylor expansion around $t+h/2$ instead that around $t$

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

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

Subtracting the second expression from the first, we get

\[
x(t + h) = x(t) + h \left(\frac{dx}{dt}\right)_{t+\frac{1}{2}h} + O(h^3)
\]

\[
= x(t) + h f \left(x \left(t + \frac{1}{2} h\right), t + \frac{1}{2} h\right) + O(h^3)
\]

The error scales as $h^3$ → is a second-order scheme

BUT THERE IS A PROBLEM HERE
ODEs. Midpoint or second-order Runge-Kutta method

PROBLEM:

\[ x(t + h) = x(t) + h \left( \frac{dx}{dt} \right)_{t + \frac{1}{2}h} + O(h^3) \]

\[ = x(t) + h f \left( x \left( t + \frac{1}{2}h \right), t + \frac{1}{2}h \right) + O(h^3) \]

This eq. requires that we know \( x(t+h/2) \) which we still do not know

IMPLICIT SCHEME: method that depends on quantities that we still need to calculate (because they refer to the next step)

vs EXPLICIT SCHEME: method depending only on quantities that we already know (because they refer to current step)
ODEs. Midpoint or second-order Runge-Kutta method

PROBLEM:

\[
x(t + h) = x(t) + h \left( \frac{dx}{dt} \right)_{t + \frac{1}{2}h} + \mathcal{O}(h^3)
\]

\[
x(t) + h f \left( x \left( t + \frac{1}{2}h \right), t + \frac{1}{2}h \right) + \mathcal{O}(h^3)
\]

This eq. requires that we know \( x(t+h/2) \) which we still do not know.

We get around this problem by approximating \( x(t + h/2) \) with the Euler method:

\[
x \left( t + \frac{h}{2} \right) = x(t) + \frac{h}{2} f(x(t), t)
\]

where \( f(x(t), t) = \frac{dx}{dt} \)

and then substituting into the above equation:

\[
k_1 = \frac{h}{2} f(x(t), t)
\]

\[
k_2 = hf \left( x(t) + k_1, t + \frac{h}{2} \right)
\]

\[
x(t + h) = x(t) + k_2
\]

which is the practical implementation of the midpoint scheme.
ODEs. Fourth-order Runge-Kutta method

We can use the same approach to go to higher order i.e.
* by using Taylor expansion
* by evaluating the ODE in several intermediate time steps

With calculations, we derive the fourth-order Runge-Kutta as

\[
\begin{align*}
  k_1 &= \frac{1}{2} h f(x, t) \\
  k_2 &= \frac{1}{2} h f \left( x + k_1, \ t + \frac{1}{2} h \right) \\
  k_3 &= h f \left( x + k_2, \ t + \frac{1}{2} h \right) \\
  k_4 &= h f \left( x + k_3, \ t + h \right)
\end{align*}
\]

\[
x(t + h) = x(t) + \frac{1}{6} \left( 2 k_1 + 4 k_2 + 2 k_3 + k_4 \right)
\]

* Errors scale as \( h^5 \)
* Fourth-order Runge-Kutta (RK4) is considered the best match
  between accuracy and not-too-complicated programming
EXERCISE:

Write a python script to implement the Euler’s method, the midpoint method and the fourth-order Runge-Kutta method. Use this script to integrate the following differential equation:

$$\frac{dx}{dt} = -x^3 + \sin t$$  \hspace{1cm} (146)

Compare the results. For a choice of initial time $t_0 = 0.0$, final time $t_{\text{fin}} = 100$, initial position $x(t_0) = 0.0$ and step-size $h = 0.4$, you should obtain something similar to Figure 41.
ODEs. EXERCISE on Euler, Midpoint, Runge-Kutta 4
ODEs. Systems of ordinary differential equations

Same approach as we have seen in the previous sections, provided that the derivatives are with respect to a single variable

\[
\frac{dx}{dt} = f_1(x, y, t) \\
\frac{dy}{dt} = f_2(x, y, t)
\]

They must be integrated in the same timestep, simultaneously, to avoid mismatch between x and y

In contrast, partial differential equations require a different treatment

\[
\frac{\partial x}{\partial t} + \frac{\partial x}{\partial s} = f_1(x, y, t, s) \\
\frac{\partial y}{\partial t} + \frac{\partial y}{\partial s} = f_2(x, y, t, s),
\]
ODEs. Second-order and higher-order ODEs

Solving second-order (or higher-order) ODEs with one variable is trivial once we know how to solve first-order ODEs.

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

Can be rewritten as a **SYSTEM of TWO FIRST-ORDER ODES**

\[
\begin{align*}
\frac{dx}{dt} &= y \\
\frac{dy}{dt} &= f \left( x, y, t \right)
\end{align*}
\]

Solve this system using the algorithms we learnt for first-order ODEs.

To solve higher ODEs, we repeat this trick till we have a system of first-order ODEs only.
ODEs. Second-order and higher-order ODEs

CLASSICAL EXAMPLE of 2\textsuperscript{nd} order ODE for astrophysicists:

equation of motion of a star in a binary system

\[
\frac{d^2 \vec{x}_i}{dt^2} = -G m_j \frac{\vec{x}_i - \vec{x}_j}{|\vec{x}_i - \vec{x}_j|^3}
\]

can be rewritten as

\[
\begin{align*}
\frac{d\vec{x}_i}{dt} &= \vec{v}_i \\
\frac{d\vec{v}_i}{dt} &= -G m_j \frac{\vec{x}_i - \vec{x}_j}{|\vec{x}_i - \vec{x}_j|^3}
\end{align*}
\]
ODEs. Astrophysical N-body problem

Integration of the equations of motion for N –bodies subject to Newton's gravity force (1687)

\[
\frac{d^2 x_i}{dt^2} = -G \sum_{j=1, j \neq i}^{N} m_j \frac{x_i - x_j}{|x_i - x_j|^3}
\]

can be split into a system of 2 first-order ODEs

\[
\begin{align*}
\frac{dv_i}{dt} &= -G \sum_{j=1, j \neq i}^{N} m_j \frac{x_i - x_j}{|x_i - x_j|^3}, \\
\frac{dx_i}{dt} &= v_i
\end{align*}
\]
ODEs. Astrophysical N-body problem

It is the first thing you need to solve to simulate a star cluster

The second thing you need is stellar evolution
ODEs. Astrophysical N-body problem

- This eq. can be solved analytically for \( N = 2 \) (Bernoulli solution, 1710).

- In 1885, a challenge was proposed, to be answered before January 21\(^{st}\) 1889, in honour of the 60th birthday of King Oscar II of Sweden and Norway:

> “Given a system of arbitrarily many mass points that attract each according to Newton’s law, under the assumption that no two points ever collide, try to find a representation of the coordinates of each point as a series in a variable that is some known function of time and for all of whose values the series converges uniformly.”

Nobody found the solution, although many participated (including Henry Poincaré).

- 1991: the mathematician Qiudong Wang found the first convergent power series solution for a generic number of bodies. However, the solution by Q. Wang is too difficult to implement and slow to convergence. Thus, everybody solves Newton’s equation numerically for \( N \geq 3 \).
ODEs. Astrophysical N-body problem

Newton’s equation can be solved with Runge-Kutta methods. For example, the Euler scheme:

\[ x(t + h) = x(t) + h \, f(x, t) \]

\[ \vec{a}_i(t) = -G \sum_{j=1, j\neq i}^{N} m_j \frac{\vec{x}_i(t) - \vec{x}_j(t)}{|\vec{x}_i(t) - \vec{x}_j(t)|^3} \]

\[ \vec{x}_i(t + h) = \vec{x}_i(t) + \vec{v}_i(t) \, h \]

\[ \vec{v}_i(t + h) = \vec{v}_i(t) + \vec{a}_i(t) \, h \]
ODEs. EXERCISE on binary star with Euler

Newton’s equation can be solved with Runge-Kutta methods. For example, the Euler scheme:

**EXERCISE:**

Write a new script to implement Euler’s method to evolve a system of two points in two dimensions ($xy$ plane), subject to gravity forces, with the following initial conditions. Initial positions of particles 1 and 2 (in the plane $xy$): $\mathbf{x} = (1.0, -1.0)$, $\mathbf{y} = (1.0, -1.0)$. Initial velocities of particles 1 and 2 (in the plane $xy$): $\mathbf{v}_x = (-0.5, 0.5)$, $\mathbf{v}_y = (0.0, 0.0)$. Let us assume that the masses are $m_1 = m_2 = 1$, and the gravity constant in our units is $G = 1$. Let us assume $t_0 = 0$, $t_{\text{fin}} = 300$ and $h = 0.01$. The result should look like the blue line in Figure 42.
ODEs. EXERCISE on binary star with Euler

Result of previous exercise is the blue line:
ODEs. Midpoint & the astrophysical N-body problem

General expression of the midpoint scheme

\[ k_1 = \frac{h}{2} f(x(t), t) \]
\[ k_2 = hf \left( x(t) + k_1, t + \frac{h}{2} \right) \]
\[ x(t + h) = x(t) + k_2 \]

How does it look like when applied to the astrophysical N-body problem?

\[ k_{1,x} = \frac{h}{2} \frac{dx_i(t)}{dt} \]
\[ k_{1,v} = \frac{h}{2} \frac{dv_i(x_i(t), t)}{dt} \]
\[ k_{2,x} = h \frac{d(x_i(t) + k_{1,x}, t + h/2)}{dt} \]
\[ k_{2,v} = h \frac{d(v_i(t) + k_{1,v}, t + h/2)}{dt} \]

\[ x_i(t + h) = x_i(t) + k_{2,x} \]
\[ v_i(t + h) = v_i(t) + k_{2,v} \]
ODEs. Midpoint & the astrophysical N-body problem

\[
k_1, x = \frac{h}{2} \frac{dx_i(t)}{dt}
\]
\[
k_1, v = \frac{h}{2} \frac{dv_i(x_i(t), t)}{dt}
\]

\[
k_2, x = h \frac{d(x_i(t) + k_1, x, t + h/2)}{dt}
\]
\[
k_2, v = h \frac{d(v_i(t) + k_1, v, t + h/2)}{dt}
\]

\[
x_i(t + h) = x_i(t) + k_2, x
\]
\[
v_i(t + h) = v_i(t) + k_2, v
\]
ODEs. Midpoint & the astrophysical N-body problem

\[ k_{1,x} = \frac{h}{2} v_i(t) \]
\[ k_{1,v} = \frac{h}{2} a_i(t) \]

\[ k_{2,x} = h \left( v_i(t) + \frac{h}{2} a_i(t) \right) \]
\[ k_{2,v} = h \frac{d}{dt} \left( v_i(t) + \frac{h}{2} a_i(t) \right) \]

\[ x_i(t + h) = x_i(t) + k_{2,x} \]
\[ v_i(t + h) = v_i(t) + k_{2,v} \]
ODEs. Midpoint & the astrophysical N-body problem

Remember that the acceleration in Newton eqs depends only on positions \((a)\) does not depend on \(v)\)

\[
k_{1,x} = \frac{h}{2} v_i(t)
\]
\[
k_{1,v} = \frac{h}{2} a_i(t)
\]

\[
k_{2,x} = h \left( v_i(t) + \frac{h}{2} a_i(t) \right)
\]
\[
k_{2,v} = h \frac{d}{dt} \left( v_i(t) + \frac{h}{2} a_i(t) \right)
\]

Writing Euler explicitly

\[
k_{2,v} = h a_i(x_i(t) + h/2 v_i(t))
\]
ODEs. Midpoint & the astrophysical N-body problem

\[ k_{1,x} = \frac{h}{2} v_i(t) \]

\[ k_{1,v} = \frac{h}{2} a_i(t) \]

\[ k_{2,x} = h \left( v_i(t) + \frac{h}{2} a_i(t) \right) \]

\[ k_{2,v} = h a_i(x_i(t) + h/2 v_i(t)) \]

\[ x_i(t + h) = x_i(t) + k_{2,x} \]

\[ v_i(t + h) = v_i(t) + k_{2,v} \]
ODEs. Midpoint & the astrophysical N-body problem

This is the most elegant form of the midpoint scheme for the N-body problem

\[
k_{1,x} = \frac{h}{2} v_i(t)
\]
\[
k_{1,v} = \frac{h}{2} a_i(t)
\]

\[
k_{2,x} = h \left[ v_i(t) + k_{1,v} \right]
\]
\[
k_{2,v} = h a_i(x_i(t) + k_{1,x})
\]

\[
x_i(t + h) = x_i(t) + k_{2,x}
\]
\[
v_i(t + h) = v_i(t) + k_{2,v}
\]
ODEs. Midpoint & the astrophysical N-body problem

In practice,

\[ k_{1,x} = \frac{h}{2} v_i(t) \]
\[ k_{1,v} = \frac{h}{2} a_i(t) \]

\[ \begin{align*}
    k_{2,x} &= h \left[ v_i(t) + k_{1,v} \right] \\
    k_{2,v} &= h a_i \left( x_i(t) + k_{1,x} \right)
\end{align*} \]

\[ \begin{align*}
    x_i(t + h) &= x_i(t) + k_{2,x} \\
    v_i(t + h) &= v_i(t) + k_{2,v}
\end{align*} \]
ODEs. EXERCISE on binary star with midpoint/ RK 4

**EXERCISE:**

Write a new script to implement the Midpoint method and/or the Runge-Kutta 4th order method to evolve a system of two points in two dimensions ($xy$ plane) described in the previous exercise. Let us assume $t_0 = 0$, $t_{\text{fin}} = 300$ and $h = 0.01$. The result should look like the red line in Figure 42 (Midpoint and Runge-Kutta 4th order cannot be distinguished by eye in this case).
ODEs. EXERCISE on binary star with midpoint/ RK 4

Result of Euler is the blue line
Result of Midpoint is the red line
ODEs. Leapfrog scheme

- a particular version of the midpoint method
- leapfrog play (Italian: la cavallina)
- similar to Euler's method but evaluated in between a time-step
ODEs. Leapfrog scheme

Most common version of leapfrog scheme is Kick – Drift – Kick (KDK) algorithm

\[ x(t), v(t), a(t) \quad v(t+\Delta t/2) \quad x(t+\Delta t), v(t+\Delta t), a(t+\Delta t) \]

\[ t \quad t+\Delta t/2 \quad t+\Delta t \]
ODEs. Leapfrog scheme

Most common version of leapfrog scheme is Kick – Drift – Kick (KDK) algorithm

\[
x(t), v(t), a(t) \quad v(t+\Delta t/2) \quad x(t+\Delta t), v(t+\Delta t), a(t+\Delta t)
\]

\[
\begin{align*}
\text{x(t), v(t), a(t)} & \quad \text{v(t+/2)} & \quad \text{x(t+Δt), v(t+Δt), a(t+Δt)} \\
\end{align*}
\]

\[
\begin{align*}
 t & \quad t+Δt/2 & \quad t+Δt \\
\end{align*}
\]

\[
v(t) \rightarrow v(t+Δt/2)
\]
ODEs. Leapfrog scheme

Most common version of leapfrog scheme is Kick – Drift – Kick (KDK) algorithm

\[ x(t), \, v(t), \, a(t) \quad \quad v(t+\Delta t/2) \quad \quad x(t+\Delta t), \, v(t+\Delta t), \, a(t+\Delta t) \]

\[ x(t) \rightarrow x(t+\Delta t) \]

\[ v(t) \rightarrow v(t+\Delta t/2) \]
ODEs. Leapfrog scheme

Most common version of leapfrog scheme is
Kick – Drift – Kick (KDK) algorithm

\[ x(t), v(t), a(t) \quad v(t+\Delta t/2) \quad x(t+\Delta t), v(t+\Delta t), a(t+\Delta t) \]

\[ t \quad t+\Delta t/2 \quad t+\Delta t \]

\[ v(t) \rightarrow v(t+\Delta t/2) \]

\[ x(t) \rightarrow x(t+\Delta t) \]

\[ v(t+\Delta t/2) \rightarrow v(t+\Delta t) \]
ODEs. Leapfrog scheme

Most common version of leapfrog scheme is Kick – Drift – Kick (KDK) algorithm

\[ x(t), v(t), a(t) \quad v(t+\Delta t/2) \quad x(t+\Delta t), v(t+\Delta t), a(t+\Delta t) \]

\[ x(t) \rightarrow x(t+\Delta t) \]

\[ v(t) \rightarrow v(t+\Delta t/2) \]

\[ v(t+\Delta t/2) \rightarrow v(t+\Delta t) \]

\[ t \quad t+\Delta t/2 \quad t+\Delta t \]

Kick + Drift + Kick (KDK) scheme
ODEs. Leapfrog scheme

Mathematically

\[ \ddot{a}_i(t) = -G \sum_{j=1, j \neq i}^N m_j \frac{\ddot{x}_i(t) - \ddot{x}_j(t)}{|\ddot{x}_i(t) - \ddot{x}_j(t)|^3}, \]

\[ \ddot{v}_i(t + \frac{h}{2}) = \ddot{v}_i(t) + \frac{h}{2} \ddot{a}_i(t) \]

\[ \dddot{x}_i(t + h) = \dddot{x}_i(t) + h \ddot{v}_i(t + \frac{h}{2}) \]

\[ \dddot{a}_i(t + h) = -G \sum_{j=1, j \neq i}^N m_j \frac{\dddot{x}_i(t + h) - \dddot{x}_j(t + h)}{|\dddot{x}_i(t + h) - \dddot{x}_j(t + h)|^3}, \]

\[ \dddot{v}_i(t + h) = \dddot{v}_i(t + \frac{h}{2}) + \frac{h}{2} \dddot{a}_i(t + h) \]
ODEs. Leapfrog scheme

In more compact form:

\[ \ddot{a}_i(t) = -G \sum_{j=1, j \neq i}^N m_j \frac{\ddot{x}_i(t) - \ddot{x}_j(t)}{|\ddot{x}_i(t) - \ddot{x}_j(t)|^3}, \]

\[ \ddot{x}_i(t + h) = \ddot{x}_i(t) + h \ddot{v}_i(t) + \frac{h^2}{2} \ddot{a}_i(t) \]

\[ \ddot{a}_i(t + h) = -G \sum_{j=1, j \neq i}^N m_j \frac{\ddot{x}_i(t + h) - \ddot{x}_j(t + h)}{|\ddot{x}_i(t + h) - \ddot{x}_j(t + h)|^3}, \]

\[ \ddot{v}_i(t + h) = \ddot{v}_i(t) + \frac{h}{2} \left[ \ddot{a}_i(t) + \ddot{a}_i(t + h) \right] \]
ODEs. Leapfrog scheme

- second-order scheme (barely)

- surprisingly accurate

- alternative version: drift-kick-drift (DKD) leapfrog scheme, in which position is evaluated at the midpoint \((t+h/2)\), then velocity is advanced to the end and finally position is recalculated to the end of the step. You can try to derive this one by yourself

- (unlike Runge-Kutta) leapfrog is time-reversal symmetric → the error on energy conservation does not grow with time

**NOTE:** A nice way to estimate how well an integrator of celestial dynamics works is to calculate the conservation of total energy and total angular momentum as a function of time during the integration
ODEs. Exercise on binary star with leapfrog

**EXERCISE:**

Write a code to implement the leapfrog scheme. Integrate the binary star in the previous exercises with the leapfrog scheme. Compare Euler’s method with the leapfrog scheme. Choose $t_0 = 0$, $t_{\text{fin}} = 300$ and $h = 0.01$. The result should look like Figure 44. Leapfrog is much better, isn’t it?
ODEs. Exercise on binary star with leapfrog

Euler versus Leapfrog

Same initial conditions: integration of a Keplerian binary
ODEs. Euler vs Leapfrog: a simple test

Energy of an N-body system

\[ E = \sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 - G \sum_{i=1}^{N} \sum_{j>i}^{N} \frac{m_i m_j}{|r_i - r_j|} \]

For a binary star, energy in the center of mass of the system

\[ E = \frac{1}{2} \frac{m_1 m_2}{(m_1 + m_2)} |v_1 - v_2|^2 - G \frac{m_1 m_2}{|r_1 - r_2|} \]

Modulus of angular momentum

\[ L = \sum_{i=1}^{N} |m_i v_i \times r_i| \]

If energy and angular momentum are supposed to be conserved in the system we simulate, the level of energy / angular momentum conservation between previous and next step is a good indicator of the accuracy of the integrator
ODEs. Euler vs Leapfrog: a simple test

Energy conservation test

Leapfrog Delta E / E ~ 2.1e-06  Euler Delta E/E = 0.0024
ODEs. Euler vs Leapfrog: a simple test

Angular momentum conservation test

Leapfrog Delta $L / L \sim 5.6e-16$  Euler Delta $L/L = 0.00013$