

Numerical Methods for Astrophysics:

ORDINARY DIFFERENTIAL EQUATIONS (ODEs)

Part 2

Michela Mapelli

ODEs. Fourth-order predictor-corrector Hermite scheme

- Hermite is $\geq 4^{\text{th}}$ order scheme
- a predictor-corrector scheme, because particle positions and velocities are first predicted at third-order level and then corrected to make the accuracy fourth-order
- define JERK: derivative of acceleration

$$\vec{j}_i \equiv \frac{d\vec{a}_i}{dt} = -G \sum_{j=1, j \neq i}^N m_j \left[\frac{\vec{v}_{ij}}{x_{ij}^3} - 3 \frac{(\vec{x}_{ij} \cdot \vec{v}_{ij}) \vec{x}_{ij}}{x_{ij}^5} \right]$$

- snap: second derivative of acceleration

$$\vec{j}_i^{\cdot}$$

- crackle: third derivative of acceleration

$$\vec{j}_i^{\cdot\cdot}$$

ODEs. Fourth-order predictor-corrector Hermite scheme

- Taylor expansions (fourth order for positions and velocities):

$$\vec{x}_i(t+h) = \vec{x}_i(t) + h\vec{v}_i(t) + \frac{h^2}{2}\vec{a}_i(t) + \frac{h^3}{6}\vec{j}_i(t) + \frac{h^4}{24}\vec{j}'_i(t) \quad (1)$$

$$\vec{v}_i(t+h) = \vec{v}_i(t) + h\vec{a}_i(t) + \frac{h^2}{2}\vec{j}_i(t) + \frac{h^3}{6}\vec{j}'_i(t) + \frac{h^4}{24}\vec{j}''_i(t) \quad (2)$$

$$\vec{a}_i(t+h) = \vec{a}_i(t) + h\vec{j}_i(t) + \frac{h^2}{2}\vec{j}'_i(t) + \frac{h^3}{6}\vec{j}''_i(t) \quad (3)$$

$$\vec{j}_i(t+h) = \vec{j}_i(t) + h\vec{j}'_i(t) + \frac{h^2}{2}\vec{j}''_i(t) \quad (4)$$

- We can use equations (3) and (4) to eliminate snap and crackle from equations (1) and (2)

$$\vec{x}_i(t+h) = \vec{x}_i(t) + \frac{h}{2} [\vec{v}_i(t) + \vec{v}_i(t+h)] + \frac{h^2}{12} [\vec{a}_i(t) - \vec{a}_i(t+h)] + \mathcal{O}(h^5) \quad \star$$

$$\vec{v}_i(t+h) = \vec{v}_i(t) + \frac{h}{2} [\vec{a}_i(t) + \vec{a}_i(t+h)] + \frac{h^2}{12} [\vec{j}_i(t) - \vec{j}_i(t+h)] + \mathcal{O}(h^5)$$

**fourth order accuracy, with errors scaling as h^5 ,
but the terms in h^3 and h^4 have canceled out!**

BUT PROBLEM HERE: we do not know $v(t+h)$, $a(t+h)$, $j(t+h)$

ODEs. Fourth-order predictor-corrector Hermite scheme

$$\begin{aligned}\vec{x}_i(t+h) &= \vec{x}_i(t) + \frac{h}{2} [\vec{v}_i(t) + \vec{v}_i(t+h)] + \frac{h^2}{12} [\vec{a}_i(t) - \vec{a}_i(t+h)] + \mathcal{O}(h^5) \\ \vec{v}_i(t+h) &= \vec{v}_i(t) + \frac{h}{2} [\vec{a}_i(t) + \vec{a}_i(t+h)] + \frac{h^2}{12} [\vec{j}_i(t) - \vec{j}_i(t+h)] + \mathcal{O}(h^5)\end{aligned}$$



- **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)

We cannot use an implicit method unless we have some trick to predict the quantities which refer to the next step.

- In the Hermite method, we predict positions and velocities of the next step by using a **third-order** Taylor expansion.

ODEs. Fourth-order predictor-corrector Hermite scheme

1- Predictor step: we use the 3rd (2nd)order Taylor expansion to predict pos (vel)

$$\vec{x}p_i(t+h) = \vec{x}_i(t) + h \vec{v}_i(t) + \frac{h^2}{2} \vec{a}_i(t) + \frac{h^3}{6} \vec{j}_i(t)$$

$$\vec{v}p_i(t+h) = \vec{v}_i(t) + h \vec{a}_i(t) + \frac{h^2}{2} \vec{j}_i(t)$$

2- Force evaluation: Calculate acceleration and jerk with the predicted pos and vel

$$\vec{a}p_i(t+h) = -G \sum_{j=1, j \neq i}^N m_j \frac{\vec{x}p_{ij}(t+h)}{xp_{ij}^3},$$

$$\vec{j}p_i(t+h) = -G \sum_{j=1, j \neq i}^N m_j \left[\frac{\vec{v}p_{ij}}{xp_{ij}^3} - 3 \frac{(\vec{x}p_{ij} \cdot \vec{v}p_{ij}) \vec{x}p_{ij}}{xp_{ij}^5} \right]$$

3- Corrector step: we substitute $vp_i(t+h)$, $ap_i(t+h)$ and $jp_i(t+h)$ to $v_i(t+h)$, $a_i(t+h)$ and $j_i(t+h)$ into the two equations marked with ★

$$\vec{x}_i(t+h) = \vec{x}_i(t) + \frac{h}{2} [\vec{v}_i(t) + \vec{v}p_i(t+h)] + \frac{h^2}{12} [\vec{a}_i(t) - \vec{a}p_i(t+h)]$$

$$\vec{v}_i(t+h) = \vec{v}_i(t) + \frac{h}{2} [\vec{a}_i(t) + \vec{a}p_i(t+h)] + \frac{h^2}{12} [\vec{j}_i(t) - \vec{j}p_i(t+h)]$$

PROBLEM HERE! Is this really 4th order?

ODEs. Fourth-order predictor-corrector Hermite scheme

PROBLEM HERE! Is this really 4th order?

$$\vec{x}_i(t+h) = \vec{x}_i(t) + \frac{h}{2} [\vec{v}_i(t) + \underline{v\vec{p}_i(t+h)}] + \frac{h^2}{12} [\vec{a}_i(t) - a\vec{p}_i(t+h)]$$

$$\vec{v}_i(t+h) = \vec{v}_i(t) + \frac{h}{2} [\vec{a}_i(t) + a\vec{p}_i(t+h)] + \frac{h^2}{12} [\vec{j}_i(t) - j\vec{p}_i(t+h)]$$

NO, first eq. contains $h v\vec{p}_i(t+h)$ which is 3rd order

TRICK: calculate first the velocity $\vec{v}_i(t+h)$ and then use it instead of $v\vec{p}_i(t+h)$

$$\underline{\vec{v}_i(t+h)} = \vec{v}_i(t) + \frac{h}{2} [\vec{a}_i(t) + a\vec{p}_i(t+h)] + \frac{h^2}{12} [\vec{j}_i(t) - j\vec{p}_i(t+h)]$$

$$\vec{x}_i(t+h) = \vec{x}_i(t) + \frac{h}{2} [\vec{v}_i(t) + \underline{\vec{v}_i(t+h)}] + \frac{h^2}{12} [\vec{a}_i(t) - a\vec{p}_i(t+h)]$$

ODEs. Fourth-order predictor-corrector Hermite scheme

SUMMARY OF THE ALGORITHM:

1- Predictor step:

$$\begin{aligned}\vec{x}_{p_i}(t+h) &= \vec{x}_i(t) + h \vec{v}_i(t) + \frac{h^2}{2} \vec{a}_i(t) + \frac{h^3}{6} \vec{j}_i(t) \\ \vec{v}_{p_i}(t+h) &= \vec{v}_i(t) + h \vec{a}_i(t) + \frac{h^2}{2} \vec{j}_i(t)\end{aligned}\tag{A}$$

2- Force evaluation:

$$\begin{aligned}\vec{a}_{p_i}(t+h) &= -G \sum_{j=1, j \neq i}^N m_j \frac{\vec{x}_{p_{ij}}(t+h)}{x_{p_{ij}}^3}, \\ \vec{j}_{p_i}(t+h) &= -G \sum_{j=1, j \neq i}^N m_j \left[\frac{\vec{v}_{p_{ij}}}{x_{p_{ij}}^3} - 3 \frac{(\vec{x}_{p_{ij}} \cdot \vec{v}_{p_{ij}}) \vec{x}_{p_{ij}}}{x_{p_{ij}}^5} \right]\end{aligned}$$

3- Corrector step:

$$\begin{aligned}\vec{v}_i(t+h) &= \vec{v}_i(t) + \frac{h}{2} [\vec{a}_i(t) + \vec{a}_{p_i}(t+h)] + \frac{h^2}{12} [\vec{j}_i(t) - \vec{j}_{p_i}(t+h)] \\ \vec{x}_i(t+h) &= \vec{x}_i(t) + \frac{h}{2} [\vec{v}_i(t) + \vec{v}_i(t+h)] + \frac{h^2}{12} [\vec{a}_i(t) - \vec{a}_{p_i}(t+h)]\end{aligned}$$

ODEs. Exercise on binary star with Hermite

EXERCISE:

Write a code to implement the Hermite scheme. Re-do the previous exercise with the Hermite scheme. Use $t_0 = 0.0$, final time $t_{\text{fin}} = 300$ and step-size $h = 0.01$. Compare the result of this exercise with the result of the leapfrog scheme. Figure 45 shows a comparison of Hermite and leapfrog also in terms of total energy conservation. The Hermite scheme performs better than the leapfrog scheme.

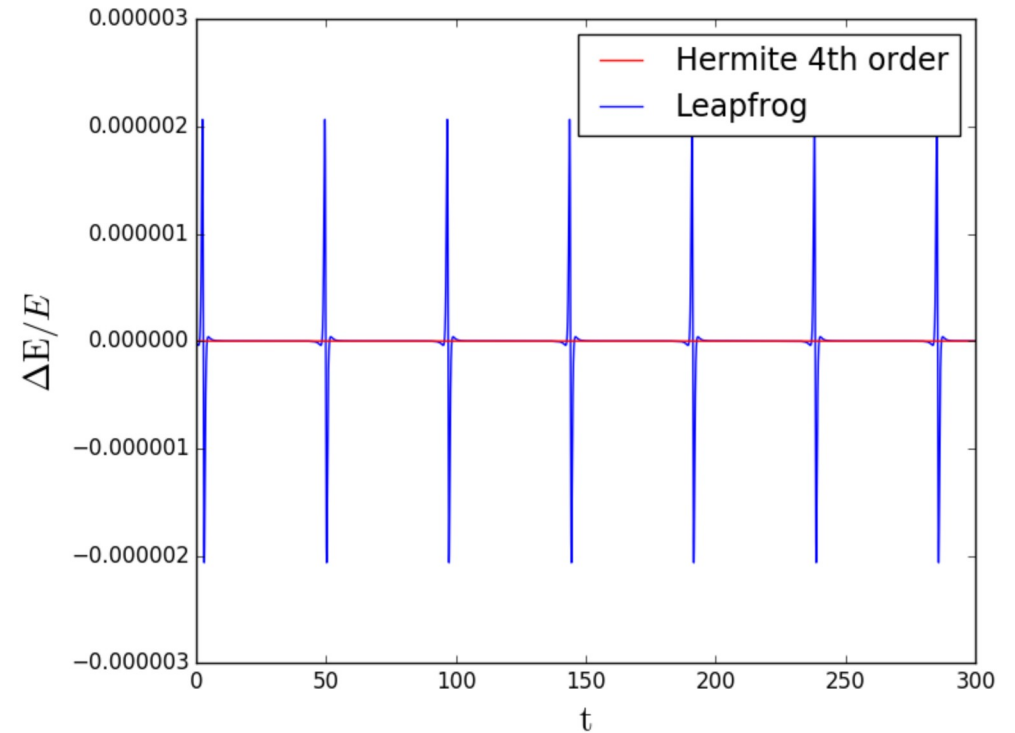
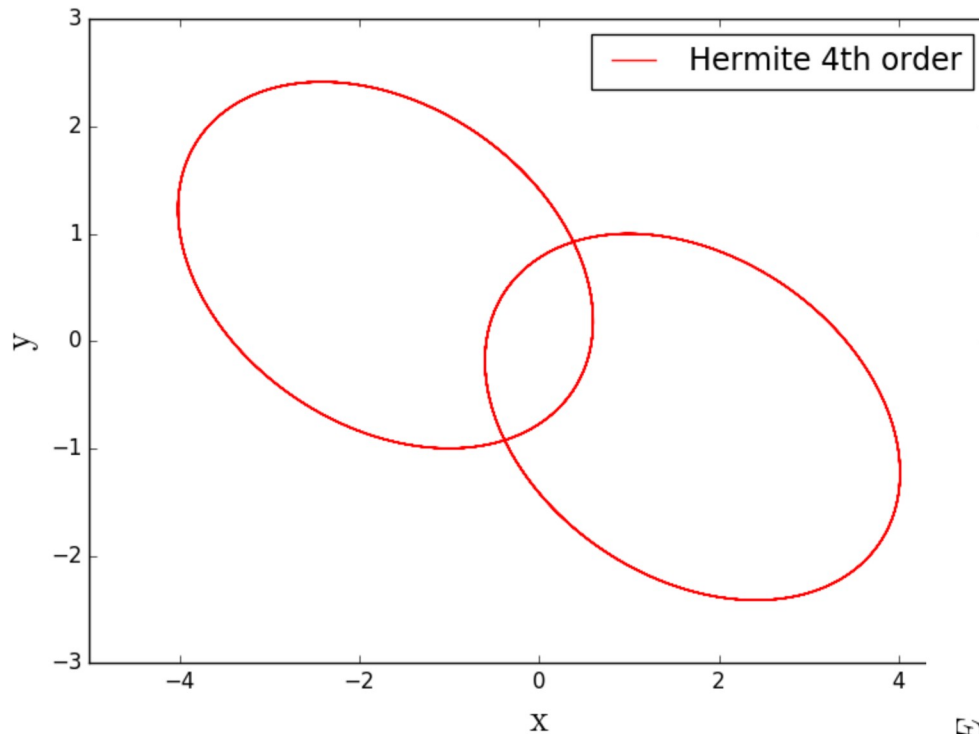
Please, don't do this exercise unless you are terribly bored by the simpler ones.

Suggestion: to calculate energy conservation, use the reference frame of the reduced particle and neglect the kinetic energy of the center of mass. It simplifies your life. In this case, the total energy of the binary is

$$E = \frac{1}{2} \frac{m_1 m_2}{(m_1 + m_2)} v_{ij}^2 - \frac{G m_1 m_2}{2 x_{ij}}, \quad (178)$$

where m_1 and m_2 are the masses of the two components of the binary star, $v_{ij} \equiv |\mathbf{v}_i - \mathbf{v}_j|$ is the modulus of the relative velocity and $x_{ij} \equiv |\mathbf{x}_i - \mathbf{x}_j|$ is the modulus of the relative distance between the two particles.

ODEs. Exercise on binary star with Hermite



Energy / ang. mom. conservation: $\Delta E/E \sim 1.5e-10$ $\Delta L/L \sim 1.2e-11$
For leapfrog was $\Delta E/E \sim 2.1e-6$ $\Delta L/L \sim 5.6e-16$

ODEs. Collisional vs collisionless N-body simulations

COLLISIONLESS SYSTEMS:

- close encounters between stars can be neglected
- the two-body relaxation timescale is longer than the Hubble time

$$t_{\text{rlx}} = \frac{N}{8 \ln N} \frac{R}{v}$$

Time for a star to completely lose memory of its initial velocity because of two-body encounters ($\Delta v \sim v$)

Galaxies and clusters of galaxies

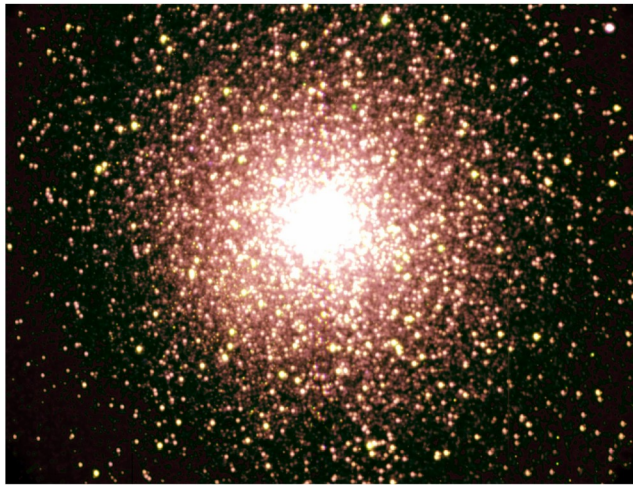


ODEs. Collisional vs collisionless N-body simulations

COLLISIONAL SYSTEMS:

- close encounters between stars drive the evolution of the system
- the two-body relaxation timescale is SHORTER than the Hubble time

Star clusters



ODEs. Collisional vs collisionless N-body simulations

COLLISIONAL SYSTEMS:

close encounters between stars are the key ingredient of the evolution

- we need DIRECT N-BODY codes (solve Newton's equation directly)
with high-order integrator (Hermite)
- predictor – corrector particularly convenient:
dynamically active stars are treated with both predictor and corrector
less dynamically active stars are treated with predictor only

COLLISIONLESS SYSTEMS:

we do not need to deal with close encounters

- other kind of N-body codes
with low-order integrators (leapfrog, midpoint,..)

ODEs. Adaptive step size

So far we have assumed $h = \text{constant}$

In general, h is not constant and should be adapted to the problem:

smaller h means high accuracy but slow computation

→ we want to choose h as a compromise
between accuracy and computing time

No universal choice of h , we must look at specific case

e.g. when simulating a stellar system

1. check that energy conservation is good enough

2. set h based on how fast the quantity we integrate changes

$$h \propto \frac{v}{a}$$

where $v = \text{velocity}$, $a = \text{acceleration}$

ODEs. When adaptive step size matters

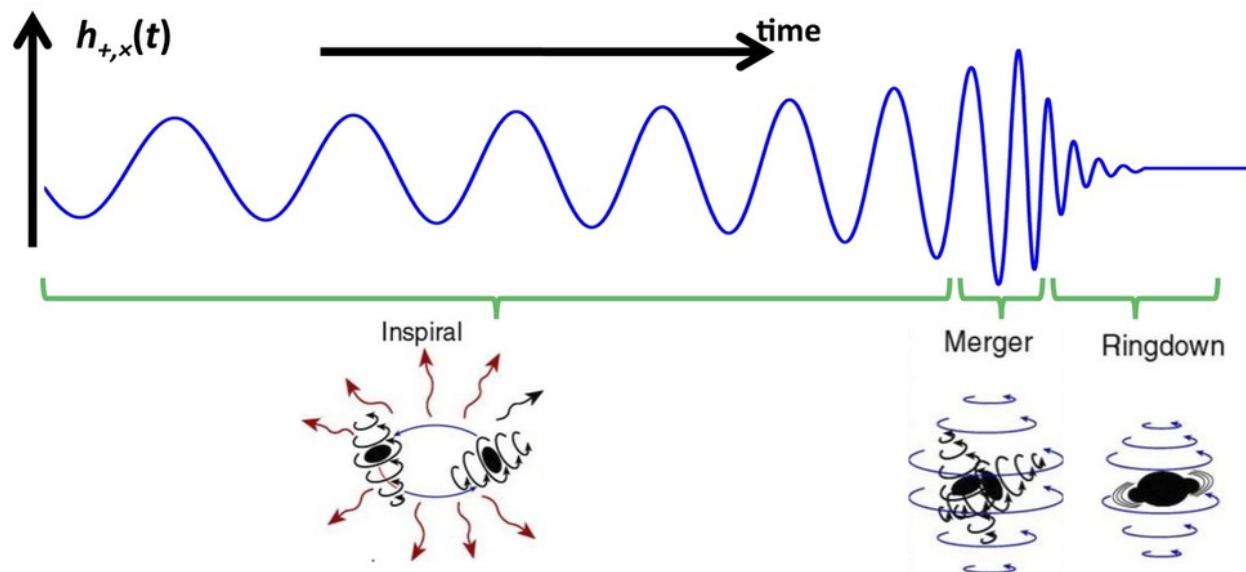
We now see an example of a system of two 1-st order ODEs when adaptive step-size is important:

EVOLUTION of the SEMI-MAJOR AXIS and of the ECCENTRICITY of a BINARY STAR because of GRAVITATIONAL-WAVE EMISSION

See Peters 1964, <https://ui.adsabs.harvard.edu/abs/1964PhRv..136.1224P/abstract>

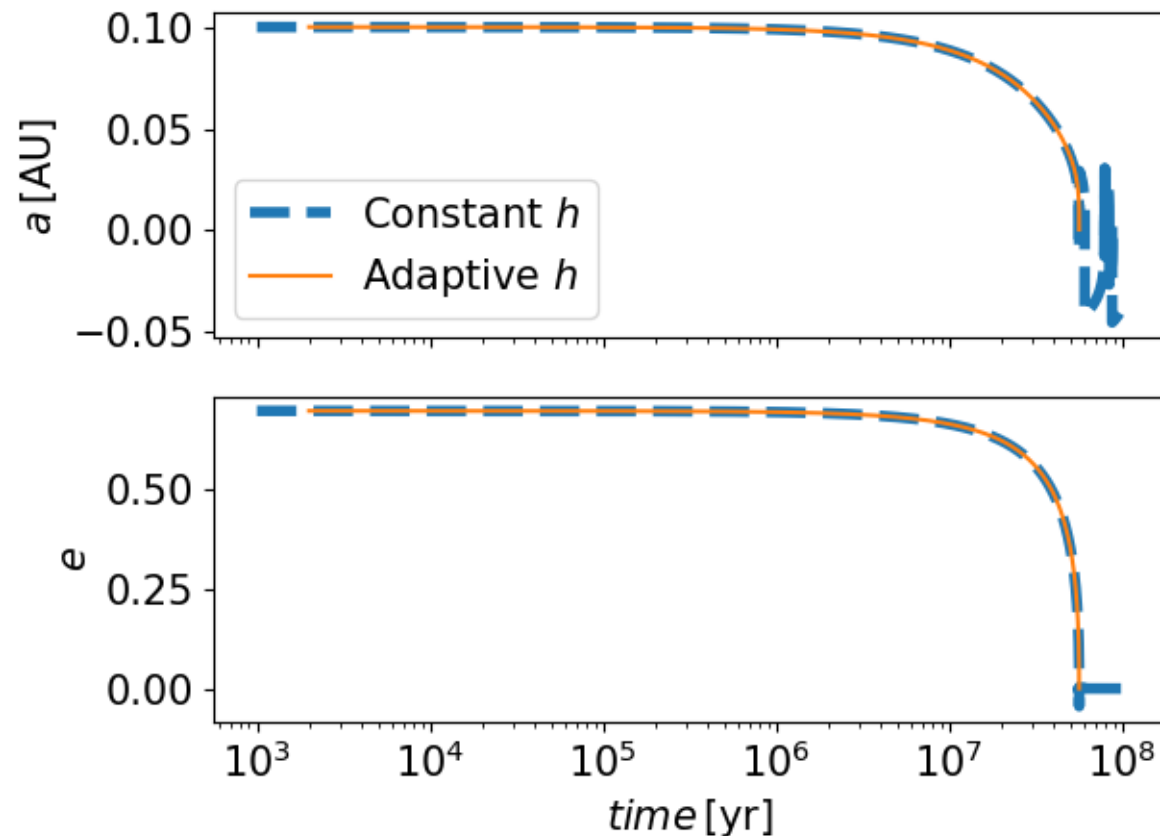
$$\frac{da}{dt} = -\frac{64}{5} \frac{G^3 m_1 m_2 (m_1 + m_2)}{c^5 a^3 (1 - e^2)^{7/2}} \left(1 + \frac{73}{24} e^2 + \frac{37}{96} e^4 \right)$$

$$\frac{de}{dt} = -\frac{304}{15} e \frac{G^3 m_1 m_2 (m_1 + m_2)}{c^5 a^4 (1 - e^2)^{5/2}} \left(1 + \frac{121}{304} e^2 \right)$$



ODEs. When adaptive step size matters

If you integrate the system of ODEs with fixed timestep, you realize that either you have a very bad result (negative a and e as soon as gravitational waves start being important, blue line below) or a terribly slow integration



ODEs. When adaptive step size matters

This problem happens is that both da/dt and de/dt are steep functions of a and e , namely

$$da/dt \propto a^{-3} (1 - e^2)^{-7/2}$$

$$de/dt \propto a^{-4} (1 - e^2)^{-5/2}$$

How can we choose an adaptive timestep in this case?

An option is that we require the relative variation of the semi-major axis to be nearly constant, or at least smaller than a chosen tolerance, during the integration.

```
tol=1e-2
h=3.1536e10 #1e3 yr
while(a>=rth):
    anew, enew=runge4(m1,m2,a,e,h)

    if(abs(anew-a)/a<(0.1*tol)): #set adaptive timestep
        h=h*2.
        anew, enew=runge4(m1,m2,a,e,h)

    elif(abs(anew-a)/a>tol):
        while(abs(anew-a)/a>tol):
            h=h/10.
            anew, enew=runge4(m1,m2,a,e,h)

    a=anew
    e=enew
    t+=h
```


ODEs. When adaptive step size matters

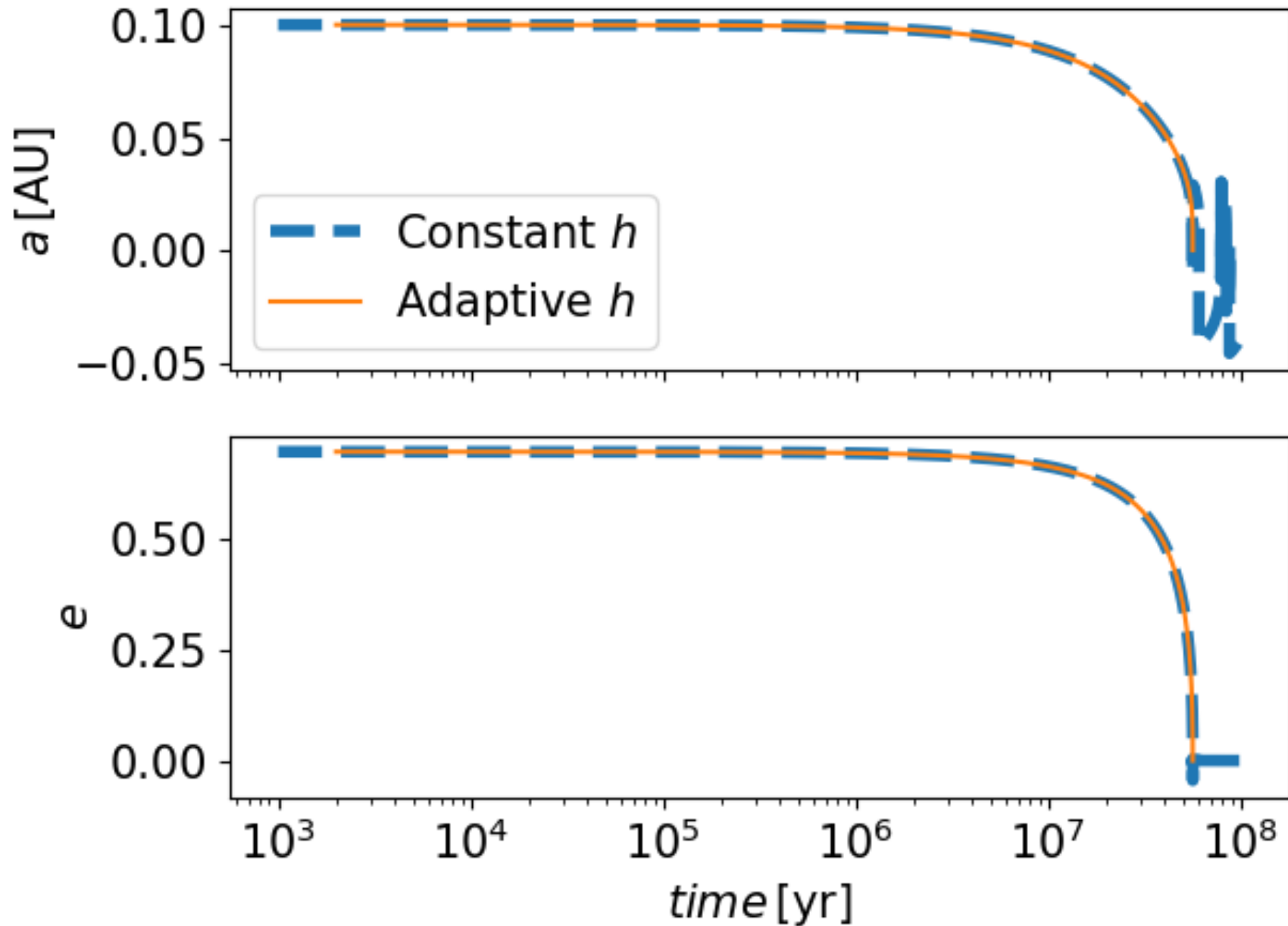
EXERCISE:

Use your Euler's script (or midpoint or RK 4th order or whatever you want) to integrate eqs. 186. Assume $m_1 = m_2 = 30 M_\odot$, $a(t = 0) = 1 \text{ A.U.}$, $e(t = 0) = 0.7$. Integrate this system till it reaches the last stable orbit of general relativity, defined as $a_{\text{LSO}} = 3 r_{\text{sch}}$, where $r_{\text{sch}} = \frac{2 G(m_1+m_2)}{c^2}$ is the Schwarzschild radius. As stopping condition for the main loop, use the following:

```
while( (abs(a-aLSO)/aLSO>1e-1) and (iterate<1e5)):
```

this means that you stop either when the difference between the semi-major axis a and the last stable orbit a_{LSO} is $< 10\%$ or when the number of iterations (iterate is a counter) is $\geq 10^5$. Start with a constant timestep $h = 10^3 \text{ yr.}$

ODEs. When adaptive step size matters



ODEs. Modified mid-point method

Let's refresh the mid-point method:

The starting point of the midpoint scheme is the following equation

$$x(t+h) = x(t) + h f\left(x\left(t + \frac{h}{2}\right), t + \frac{h}{2}\right), \quad (187)$$

which is implicit because of $x\left(t + \frac{h}{2}\right)$. In the original version of the midpoint, we have used Euler's method to calculate $x\left(t + \frac{h}{2}\right) = x(t) + \frac{h}{2} f(x, t)$:

$$x(t+h) = x(t) + h f\left(x(t) + \frac{h}{2} f(x, t), t + \frac{h}{2}\right) \quad (188)$$

After this first step, in the usual midpoint scheme we calculate the midpoint of the next interval:

$$x\left(t + \frac{3}{2}h\right) = x(t+h) + \frac{h}{2} f(x(t+h), t+h) \quad (189)$$

ODEs. Modified mid-point method

Here I just repeat the last equation of slide 19

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

An alternative to the above expression is to evaluate directly the midpoint of the next timestep from the midpoint of the previous timestep:

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

And then calculate the full timestep as

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

and we repeat this process as many times as we want.

Hence, the general expression of this new scheme is:

$$x(t + h) = x(t) + h f\left(x(t) + \frac{h}{2} f(x, t), t + \frac{h}{2}\right)$$
$$x\left(t + \frac{3}{2}h\right) = \left[x(t) + \frac{h}{2} f(x, t)\right] + h f(x(t + h), t + h).$$

Suppose that we now want to solve an ODE from t to $t + H$ using n steps of size $h = H/n$. Let us write down the above equations in a still different form.

ODEs. Modified mid-point method

Suppose that we now want to solve an ODE from t to $t + H$ using n steps of size $h = H/n$. Let us write down the above equations in a still different form.

The first half step is always:

$$x_0 = x(t)$$
$$y_1 = x_0 + \frac{1}{2}h f(x_0, t)$$

Then, the next n steps are:

$$x_1 = x_0 + h f\left(y_1, t + \frac{1}{2}h\right)$$
$$y_2 = y_1 + h f(x_1, t + h)$$
$$x_2 = x_1 + h f\left(y_2, t + \frac{3}{2}h\right)$$
$$y_3 = y_2 + h f(x_2, t + 2h)$$

...
The x_i terms are the solutions at integer multiples of h ,
The y_i are the solutions at half-integer multiples.

Hence we can write the equations in a more compact form as:

$$y_{m+1} = y_m + h f(x_m, t + m h)$$
$$x_{m+1} = x_m + h f\left(y_{m+1}, t + \left(m + \frac{1}{2}\right) h\right)$$

ODEs. Modified mid-point method

The best way to estimate the final point (as shown in the Lecture notes) is

$$x(t + H) = \frac{1}{2} \left[x_n + y_n + \frac{1}{2} h f(x_n, t + H) \right]$$

It can be shown that all the error terms containing odd powers of h that arise from the Euler's method step at the start of the calculation cancel out, giving a **total error that contains only even powers of h** .

SUMMARY of the MODIFIED MID-POINT METHOD:

$$\begin{aligned} x_0 &= x(t) \\ y_1 &= x_0 + \frac{1}{2} h f(x_0, t) \\ x_1 &= x_0 + h f\left(y_1, t + \frac{1}{2} h\right) \\ y_{m+1} &= y_m + h f(x_m, t + m h) \quad \forall 1 \leq m \leq (n - 1) \\ x_{m+1} &= x_m + h f\left(y_{m+1}, t + \left(m + \frac{1}{2}\right) h\right) \quad \forall 1 \leq m \leq (n - 1) \\ x(t + H) &= \frac{1}{2} \left[x_n + y_n + \frac{1}{2} h f(x_n, t + H) \right] \end{aligned}$$

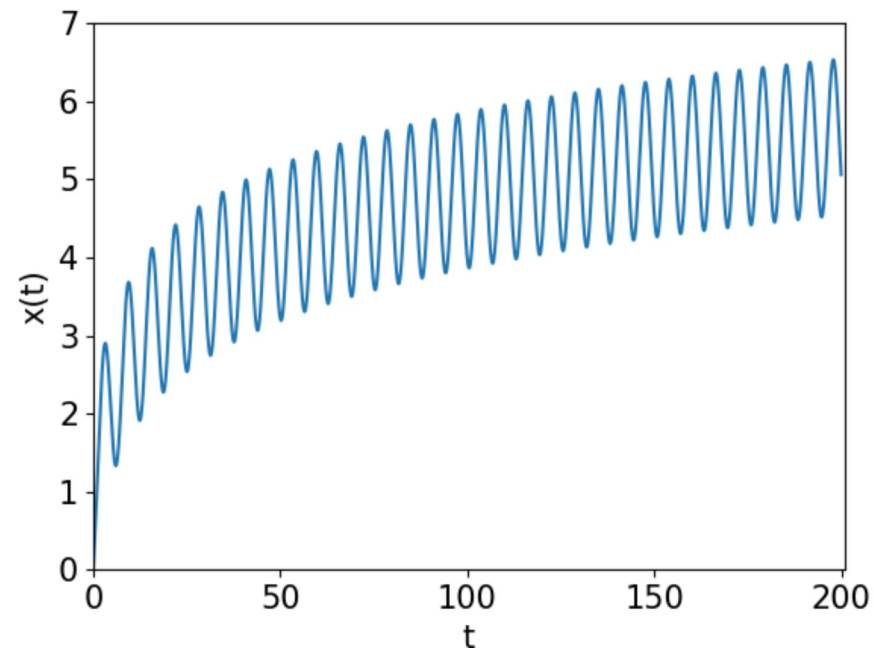
ODEs. Modified mid-point method

EXERCISE:

Write a script to implement the modified midpoint method. Use it to evolve the following differential equation

$$\frac{dx}{dt} = \exp(-x) + \sin(t) \quad (200)$$

Let us assume $t_0 = 0$, $t_{\text{fin}} = 200$, $h = 0.01$ and $x_0 = 0$. The result should look like Figure 47.



ODEs. Bulirsch – Stoer method

The Bulirsch-Stoer method exploits the advantages of the modified midpoint method to reach an **accuracy as large as we want, at least in principle**

Let's consider, for simplicity, a first-order ODE in a single variable: $dx/dt = f(x, t)$.

We want to integrate it from t to $t + h_1$.

We start calculating the solution

- i) with the **modified midpoint method**,
- ii) in just one single timestep of size h_1 equal to the entire range.

This yields a very approximate solution $x(t + h_1) = R_{1,1}$

Then, we split the time step in two: $h_2 = h_1 / 2$

and we repeat the calculation, getting a solution $x(t + h_1) = R_{2,1}$

Since the error on the modified midpoint method is always and even function of the stepsize, we have that

$$(1) \quad x(t + h_1) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4)$$


$$(2) \quad x(t + h_1) = R_{1,1} + c_1 h_1^2 + \mathcal{O}(h_1^4) = R_{1,1} + 4 c_1 h_2^2 + \mathcal{O}(h_2^4)$$


where $c_1 =$ unknown constant,

$R_{1,1}$ and $R_{2,1}$ are the solutions with the modified midpoint method

ODEs. Bulirsch – Stoer method

We use (1) and (2) to calculate c_1 : $c_1 h_2^2 = \frac{1}{3}(R_{2,1} - R_{1,1})$


Substituting in (1): $x(t + h_1) = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1}) + \mathcal{O}(h_2^4)$ 
where we got rid of the terms in h^2

Let's call the new solution: $R_{2,2} = R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1})$ 

We can take this approach further. Let us now consider $h_3 = 1/3 h_1$.
The solution of the modified midpoint will be $x(t + h_1) = R_{3,1}$

$$x(t + h_1) = R_{3,1} + c_1 h_3^2 + \mathcal{O}(h_3^4)$$

$$x(t + h_1) = R_{2,1} + c_1 h_2^2 + \mathcal{O}(h_2^4) = R_{2,1} + \frac{9}{4}c_1 h_3^2 + \mathcal{O}(h_3^4)$$


 $c_1 h_3^2 = \frac{4}{5}(R_{3,1} - R_{2,1})$

$$R_{3,2} = R_{3,1} + \frac{4}{5}(R_{3,1} - R_{2,1})$$

$$x(t + h_1) = R_{3,2} + \mathcal{O}(h_3^4)$$


third order algorithm with fourth order errors

ODEs. Bulirsch – Stoer method

We now write: $x(t + h_1) = R_{3,2} + c_2 h_3^4 + \mathcal{O}(h_3^6)$ 

Combining the two equations indicated with 

we have

$$x(t + h_1) = R_{2,2} + c_2 h_2^4 + \mathcal{O}(h_2^6) = R_{2,2} + \left(\frac{3}{2}\right)^4 c_2 h_3^4 + \mathcal{O}(h_3^6) $$

From the equations marked with  and  we derive $c_2 h_3^4 = \frac{16}{65}(R_{3,2} - R_{2,2})$

Substituting this into  we obtain

$$x(t + h_1) = R_{3,3} + \mathcal{O}(h_3^6) \quad \text{Where} \quad R_{3,3} = R_{3,2} + \frac{16}{65}(R_{3,2} - R_{2,2})$$

The error is of order h^6 and we have taken only 3 modified midpoint steps.

The power of this method is that it cancels out the error terms to higher and higher orders on successive steps.

Let's now generalize the algorithm

ODEs. Bulirsch – Stoer method

Let's now generalize the algorithm:

- denote the current number of steps by n
- denote our modified midpoint estimate of $x(t + h_1)$ by $R_{n,1}$

We can generalize as

$$x(t + h_1) = R_{n,m} + c_m h_n^{2m} + \mathcal{O}(h_n^{2m+2}) \quad \text{😊}$$

where c_m is the unknown constant

The corresponding estimate at step $n - 1$ is

$$x(t + h_1) = R_{n-1,m} + c_m h_{n-1}^{2m} + \mathcal{O}(h_{n-1}^{2m+2}) \quad \text{😬}$$

Since $h_n = h_1/n$, $h_{n-1} = h_1/(n - 1)$ we have that $h_{n-1} = \frac{n}{n - 1} h_n$ 😊

Combining the three equations marked by 😊 and expressing in terms of c_m we find

$$c_m h_n^{2m} = \frac{R_{n,m} - R_{n-1,m}}{[n/(n - 1)]^{2m} - 1} \quad \text{🔵}$$

Finally, substituting eq. 🔵 into eq. 😊, we obtain the general Bulirsch-Stoer method

ODEs. Bulirsch – Stoer method

General formulation of the Bulirsch – Stoer method:

$$x(t + h_1) = R_{n,m+1} + \mathcal{O}(h_n^{2m+2})$$

where

$$R_{n,m+1} = R_{n,m} + \frac{R_{n,m} - R_{n-1,m}}{[n/(n-1)]^{2m} - 1}$$

Example of a binary system: [examples/ODEs/bulirsch_stoer.py](#)

ODEs. Initial-value and boundary-value problems

Initial-value problems:

we solve differential equations given the initial values of the variables

Boundary-value problems:

we do not know the initial values of the variables,
but only the values of the variables at “some point/time”.

MORE DIFFICULT to integrate

Example: we want to integrate the height above the ground of a ball thrown in the air VERTICALLY and subject to gravity force (no air friction)

$$\frac{d^2x}{dt^2} = -g$$

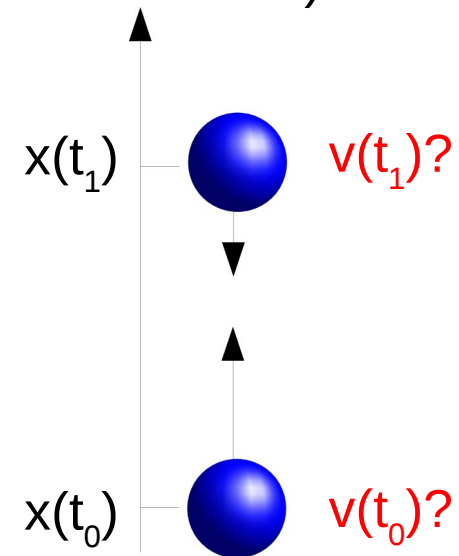
where $g = 9.81 \text{ m/s}^2$

initial-value problem:

we know initial $x = x(t_0)$ and initial $v = v(t_0)$

boundary-value problem: we know e.g.

position at time t_0 and time t_1 $x(t_0)$ and $x(t_1)$



ODEs. Initial-value and boundary-value problems

Shooting method for boundary-value problems:

we start with a guess of the initial values we do not know and then we improve our guess iteratively.

e.g. in the example of the ball thrown in the air

If we know position at t_0 and t_1 : $x_{\text{true}}(t_0)$ and $x_{\text{true}}(t_1)$

- we guess v_0 at time t_0

- we calculate $x(t_1)$

- If $x(t_1) \neq x_{\text{true}}(t_1)$,

then we come back to the initial conditions

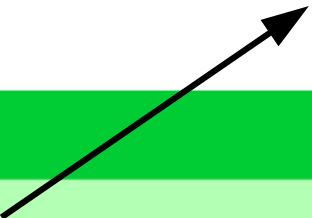
and we try with a different velocity,

namely with a larger (smaller) v_0

if we find a value of $x(t_1) < x_{\text{true}}(t_1)$ ($> x_{\text{true}}(t_1)$)

- We repeat till $|x(t_1) - x_{\text{true}}(t_1)| < \text{tolerance}$

ODEs. Exercise on shooting method

$$\frac{d^2x}{dt^2} = -g$$


EXERCISE:

Solve the problem proposed in equation 222, i.e. integrate the motion of a ball vertically thrown in the air with initial position $x(t = 0) = 0$ and position $x(t = 3 \text{ s}) = 10 \text{ m}$, at $t = 3 \text{ s}$. Neglect the viscous friction from the air. Use the shooting method.

Suggestion: use the Euler method to solve the ODEs. Take $N = 10^3$ steps between $t = 0$ and $t_f = 3 \text{ s}$. Assume a tolerance $\epsilon = 10^{-3}$ and stop the calculation when $|x(t_f) - x_{\text{true}}(t_f)| < \epsilon$, where $x_{\text{true}}(t_f) = 10 \text{ m}$.

The result should look like Figure 48. In particular, the correct initial velocity is $v_0 \sim 18.0 \text{ m/s}$.

ODEs. Exercise on shooting method

