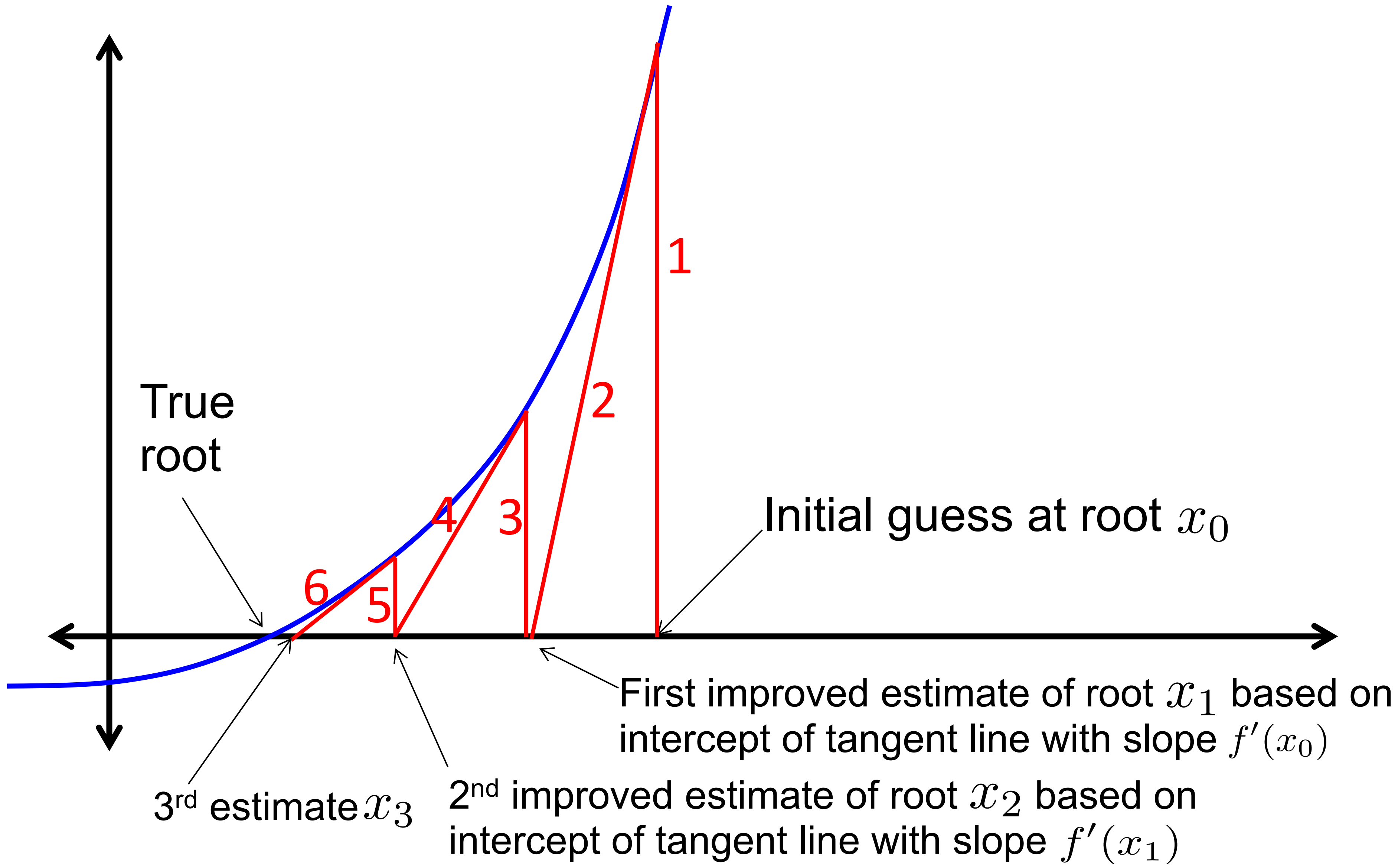
PHY604 Lecture 8

September 26, 2021

Review: Geometrical Interpretation of Newton-Raphson Iteration


True root
Initial guess at root x_0
First improved estimate of root x_1 based on intercept of tangent line with slope $f'(x_0)$
2nd improved estimate of root x_2 based on intercept of tangent line with slope $f'(x_1)$
3rd estimate x_3
1
2
3
4
5
6

Review: Pseudocode of Newton-Raphson Algorithm

- 1. Choose initial guess at the root (x_0), and the convergence tolerance (ε).
- 2. Loop through *n* up to a maximum number $N_{\max}$ (exit and tell the user that the root finding has failed if it reaches $N_{\max}$)
- 3. Make sure $f'(x) \neq 0$
- 4. Compute new estimate of root: $x_n \simeq x_{n-1} - \dfrac{f(x_{n-1})}{f'(x_{n-1})}$
- 5. Check convergence criteria:

$$|x_{n+1} - x_n| < \begin{cases} \epsilon |x_n|, & \text{when } |x_n| \neq 0 \\ \epsilon, & \text{when } |x_n| = 0 \end{cases}$$

Review: Summary of root-finding methods

- Bisection:
  - Robust (with appropriate initial guesses)
  - Slow, each iteration reduces error by a factor of two
  - Need to make sure root is within initial guesses
- Newton-Raphson:
  - Fast: often only takes a few iterations
  - Need to know derivative of function, and they must exist
  - Can diverge, e.g., in cases with small second derivatives
- Secant method
  - Similar convergence speed as NR method
  - Don’t need analytical derivatives
  - Same divergence properties as NR method
  - Numerical derivatives may be noisy

Review: Differential equations (Newman Ch. 8)

- One of the major applications of computation to science and engineering is solving differential equations
  - Even for very simple-looking equations if they are "nonlinear," they are difficult or imposible to solve analytically
- Classifications:
  - Initial value problems
  - Boundary value problems
  - Eigenvalue problems
- Often problems are described by systems of coupled differential equations
- As with the other topics, there are many different methods
  - We just want to see the basic ideas and popular methods

Review: Coupled systems of ODEs in vector notation

- In order to simplify the description of the second order Runge-Kutta algorithm we use the following vector notation to simplify the equations:

$$\mathbf{y} \equiv (y_1, y_2, y_3, \ldots, y_N)$$
$$\mathbf{f} \equiv (f_1, f_2, f_3, \ldots, f_N)$$

- Using this notation, the original set of ODEs is:

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}, t)$$

- In this notation Euler's method is:

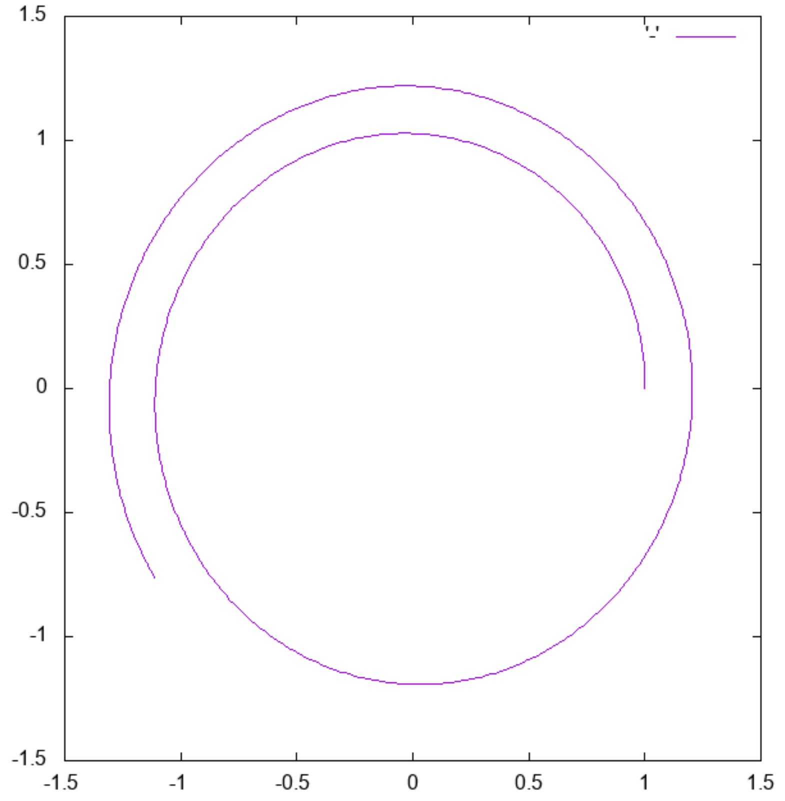
$$\mathbf{y}^{n+1} = \mathbf{y}^n + \Delta t \mathbf{f}(\mathbf{y}^n, t^n)$$

Today's lecture: More on ordinary differential equations

- Orbit example for Euler method
- Runge-Kutta and adaptive RK method
- Beyond RK: Other methods for ODEs
  - Leapfrog/Verlet/modified midpoint
  - Bulirsch-Stoer Method
- Boundary Value problems
- Eigenvalue problems

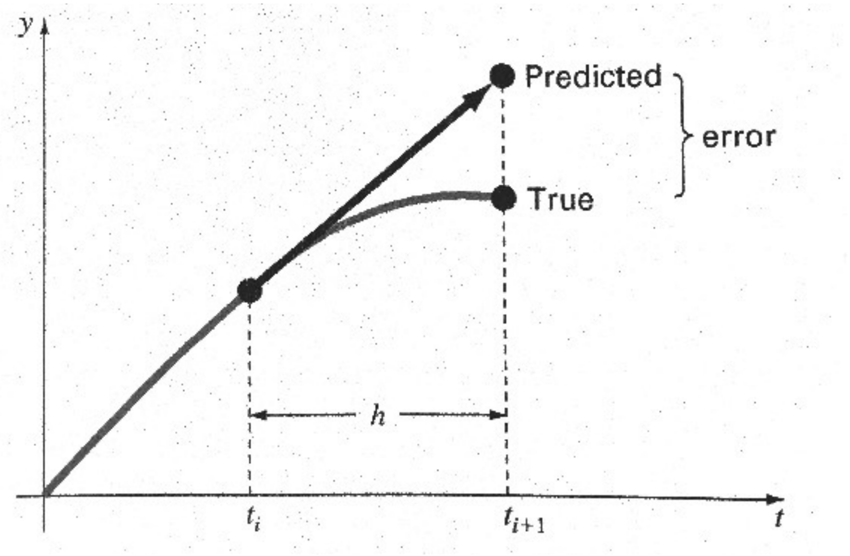
Example program for Euler orbit problem

- See **`euler_orbit.ipynb`**

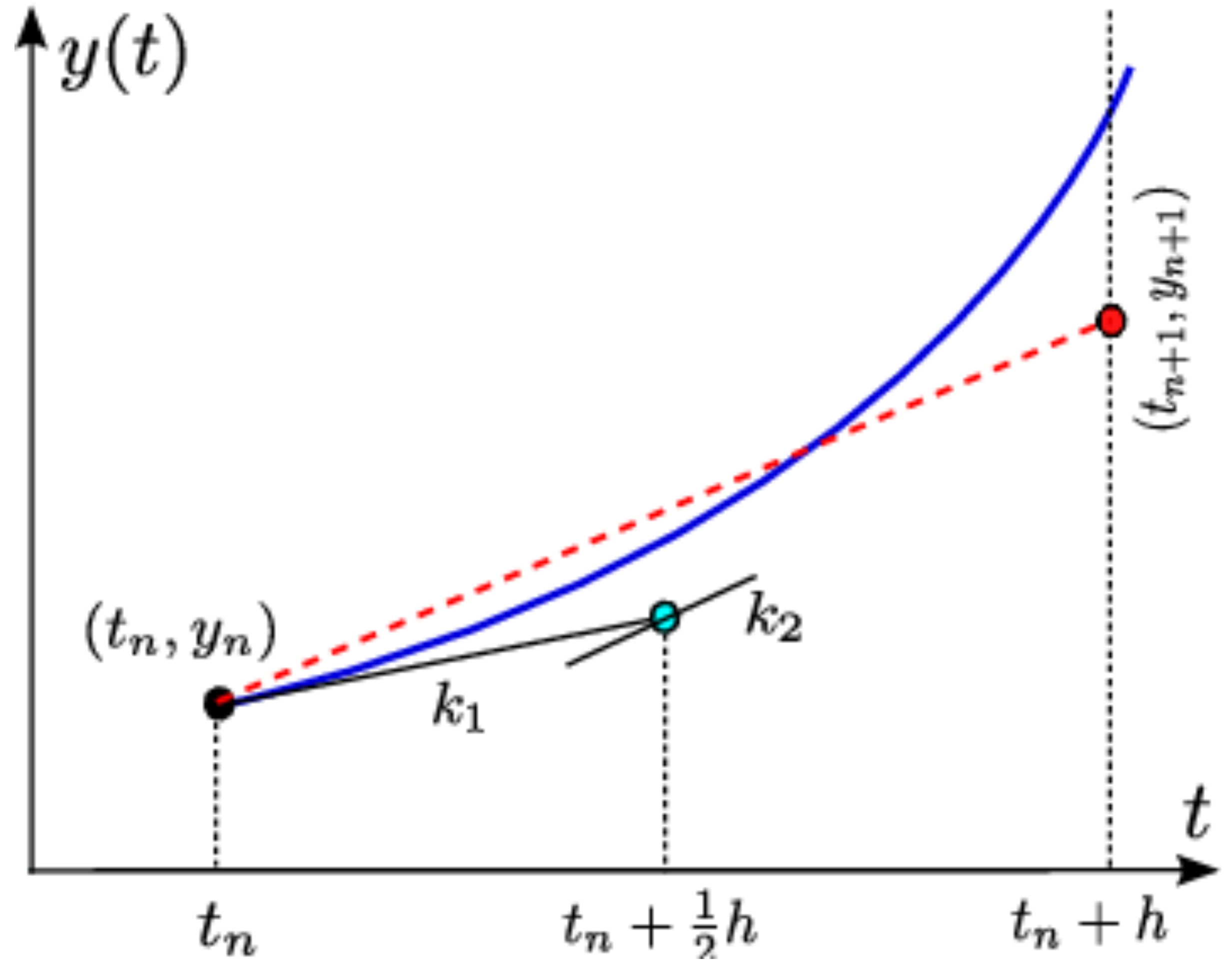


More accurate ODE numerical methods

- The problem with Euler's method is that the right-hand-side of the equations is evaluated at the beginning of the timestep
- The right-hand-side usually changes over the course of each timestep and we may be getting an inaccurate answer as a result
    - It would be better if we could evaluate the right-hand-side in the middle of the timestep.
    - However, we can't do that unless we know the solution in advance
- We could use higher-order finite differences, however this is not a common approach
- **Strategy:** Use Euler's method to estimate the solution at the midpoint of the timestep. And then use this estimate to evaluate the right-hand-side
- This is called a second order Runge-Kutta method



Second-order Runge-Kutta method



Fadlisyah, Muhammad thesis (2014)

Second-order Runge-Kutta method

- Taylor expand around *t* + 1/2 Δ*t* :

$$y(t+\Delta t) = y(t+\frac{1}{2}\Delta t) + \frac{1}{2}\Delta t \left.\frac{dy}{dt}\right|_{t+\frac{1}{2}\Delta t} + \frac{1}{8}\Delta t^2 \left.\frac{d^2y}{dt^2}\right|_{t+\frac{1}{2}\Delta t} + \mathcal{O}(\Delta t^3)$$

$$y(t) = y(t+\frac{1}{2}\Delta t) - \frac{1}{2}\Delta t \left.\frac{dy}{dt}\right|_{t+\frac{1}{2}\Delta t} + \frac{1}{8}\Delta t^2 \left.\frac{d^2y}{dt^2}\right|_{t+\frac{1}{2}\Delta t} - \mathcal{O}(\Delta t^3)$$

- Subtract the two expressions

$$y(t+\Delta t) = y(t) + \Delta t \left.\frac{dy}{dt}\right|_{t+\frac{1}{2}\Delta t} + \mathcal{O}(\Delta t^3)$$

Need *f* evaluated at midpoint

$$= y(t) + \Delta t\, f(y(t+\frac{1}{2}\Delta t), t+\frac{1}{2}\Delta t) + \mathcal{O}(\Delta t^3)$$

Second-order Runge-Kutta method

- **Step 1:** Estimate change due of the right-hand side using Euler's method:

$$\mathbf{k}_1 = \Delta t \mathbf{f}(\mathbf{y}^n, t^n)$$

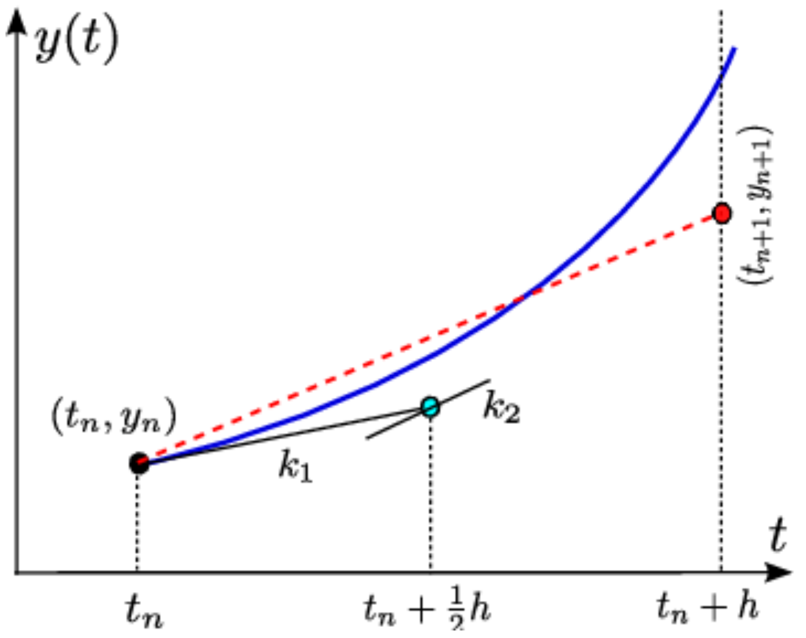
- **Step 2:** Use estimate to predict value of solution at midpoint of the timestep. Evaluate right hand side at midpoint:

$$\mathbf{y}^{n+1} = \mathbf{y}^n + \Delta t \mathbf{f}(\mathbf{y}^n + \frac{1}{2}\mathbf{k}_1, t^n + \frac{1}{2}\Delta t)$$

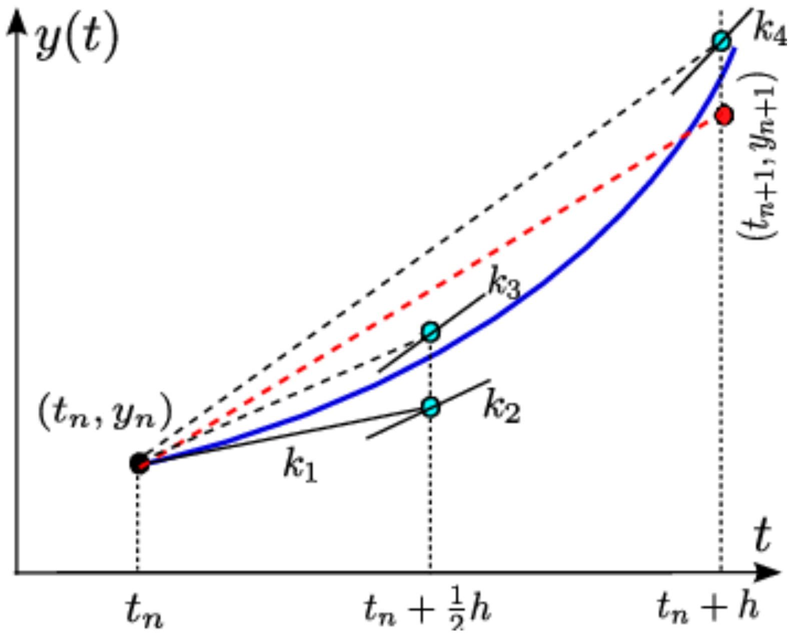
- See **`rk2_orbit.f08`**

Second and fourth-order Runge-Kutta methods

2nd order



4th order



Fadlisyah, Muhammad thesis (2014)

The fourth-order Runge-Kutta method

- In practice, the workhorse algorithm for first-order sets of ODEs is the fourth-order Runge-Kutta algorithm which (we state here without derivation)
- Step 1: $$\mathbf{k}_1 = \Delta t \ \mathbf{f}(\mathbf{y}^n, t^n)$$
- Step 2: $$\mathbf{k}_2 = \Delta t \ \mathbf{f}(\mathbf{y}^n + \frac{1}{2}\mathbf{k}_1, t^n + \frac{1}{2}\Delta t)$$
- Step 3: $$\mathbf{k}_3 = \Delta t \ \mathbf{f}(\mathbf{y}^n + \frac{1}{2}\mathbf{k}_2, t^n + \frac{1}{2}\Delta t)$$
- Step 4: $$\mathbf{k}_4 = \Delta t \ \mathbf{f}(\mathbf{y}^n + \mathbf{k}_3, t^n + \Delta t)$$
- Step 5: $$\mathbf{y}^{n+1} = \mathbf{y}^n + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

Runge-Kutta methods

- Euler method can be thought of as the first-order RK method
  - Accurate to first order in Δt, i.e., error is order Δt^2

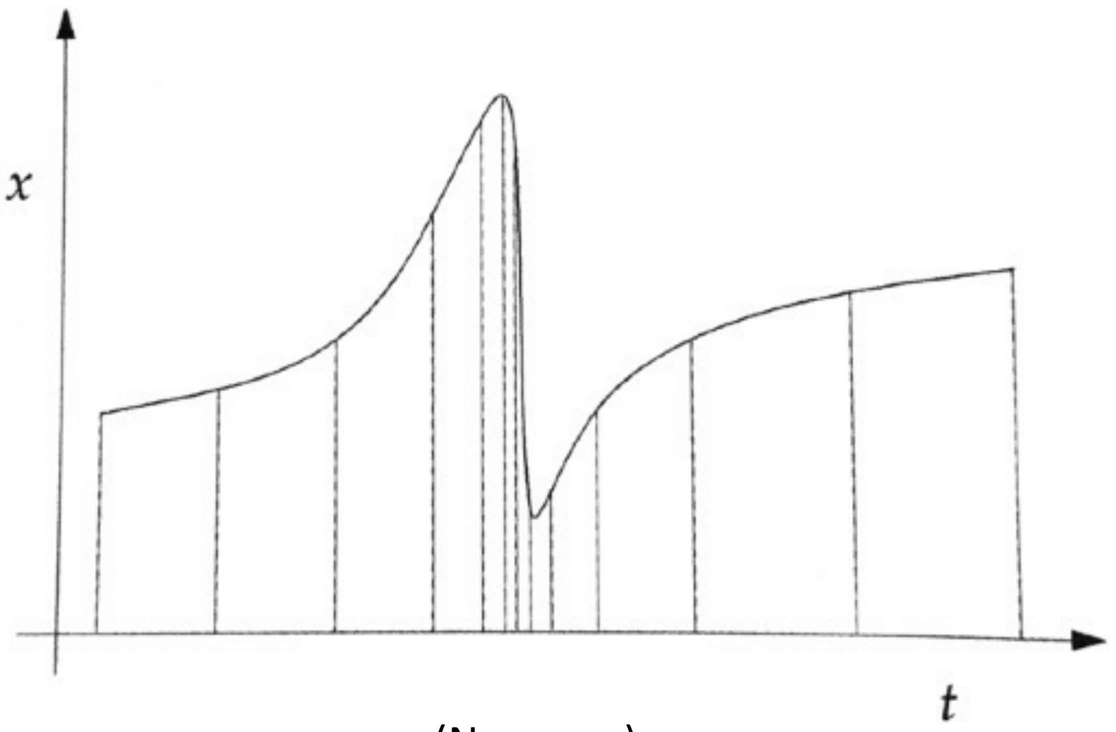
- Second-order RK method accurate to Δt^2, so error Δt^3

- Fourth-order RK method accurate to Δt^4, so error Δt^5
  - By far the most common method for the numerical solution of ODEs
  - Balances accuracy and complexity

- Quoted accuracies are for one step, errors accumulate over the number of steps needed in the calculation, usually loose an order of accuracy (see Newman)

Adaptive step size

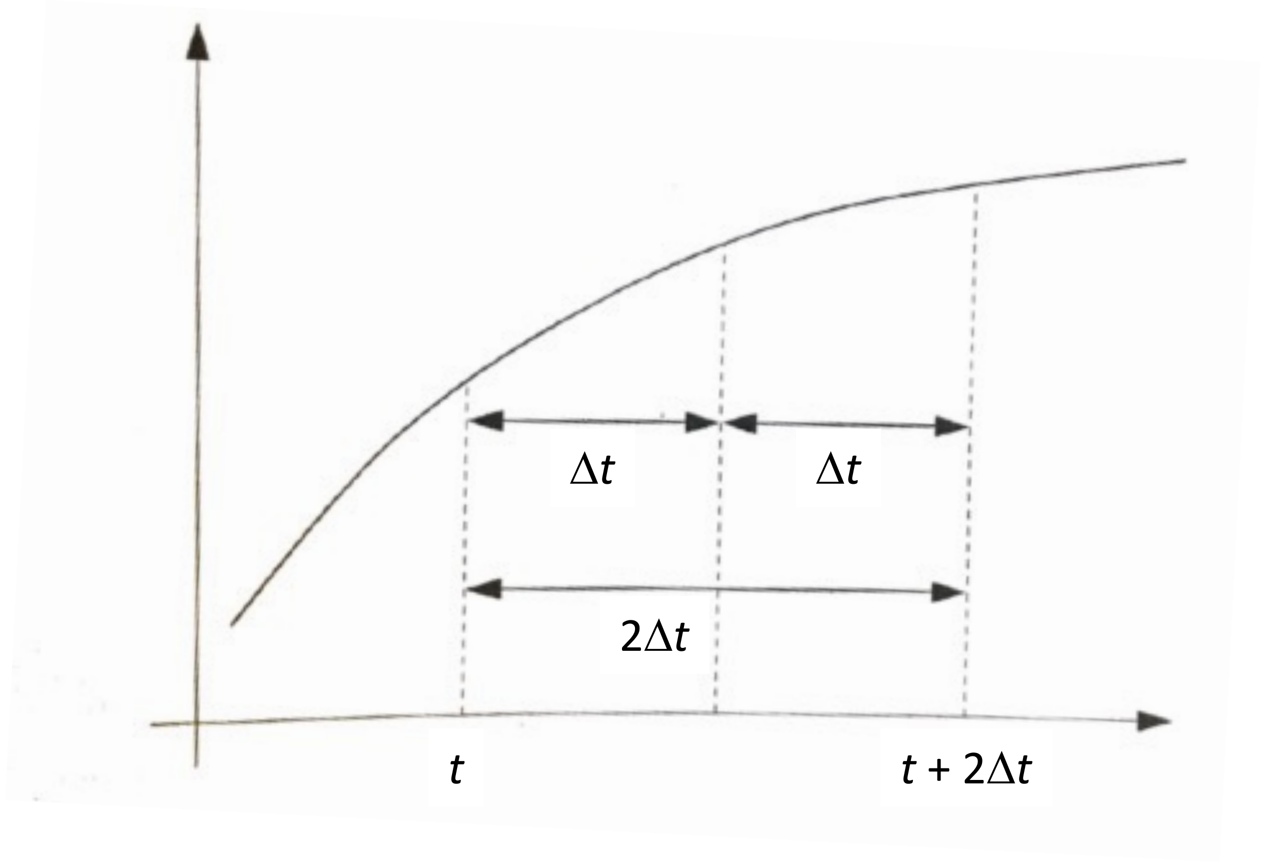
- So far, we have set by hand a constant step size Δt
- Often, we can get better results by varying the step size
  - Increase in regions where function varies rapidly, decrease where it varies slowly
- Approach: vary Δt so the error introduced per unit interval is roughly constant
  - First we need to estimate the error in the steps



(Newman)

Adaptive step size: Estimating the error

- 1. Choose initial (small) Δt
- 2. Use RK method to do two Δt steps of the solution
- 3. Go back to initial t and do an RK step with $2\Delta t$
- 4. Compare the results to estimate the error



Adaptive step size: Estimating the error

- True value of function related to estimate $y_{\Delta t}$:

$$y(t + 2\Delta t) = y_{\Delta t} + 2c\Delta t^5$$

- For doubled step size $y_{2\Delta t}$:

$$y(t + 2\Delta t) = y_{2\Delta t} + 32c\Delta t^5$$

- So per step error is:

$$\epsilon = c\Delta t^5 = \frac{1}{30}(y_{\Delta t} - y_{2\Delta t})$$

- Take δ to be the target accuracy per step. Then the step size necessary to get that accuracy is:

$$\Delta t' = \Delta t \sqrt[5]{\frac{30\delta}{|y_{\Delta t} - y_{2\Delta t}|}}$$

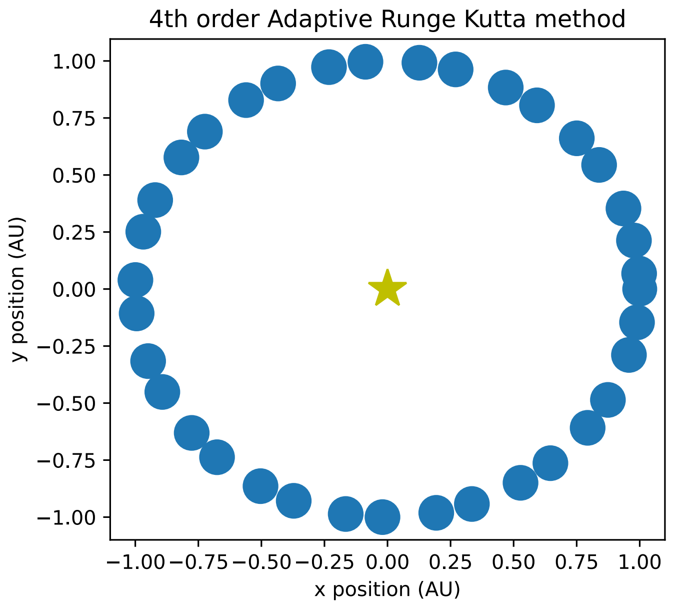
Adaptive step size: Complete approach

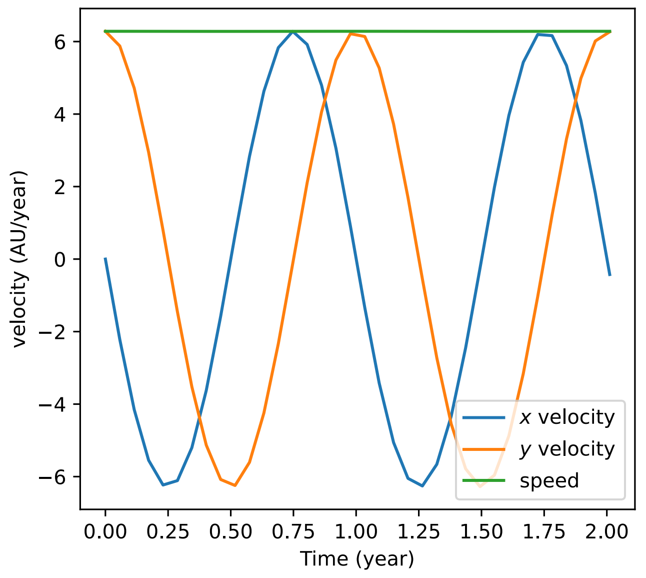
- 1. Choose initial Δt
- 2. Use RK method to do two Δt steps of the solution
- 3. Go back to initial t and do an RK step with $2\Delta t$
- 4. Compare the results to estimate the error
- 5. Calculate ideal step size $\Delta t'$
  - If $\varepsilon > \delta$, then redo the calculation with $\Delta t'$
  - If $\varepsilon < \delta$, take the results obtained using Δt and move on to time $t + \Delta t$. In the next iteration use $\Delta t'$ as the timestep

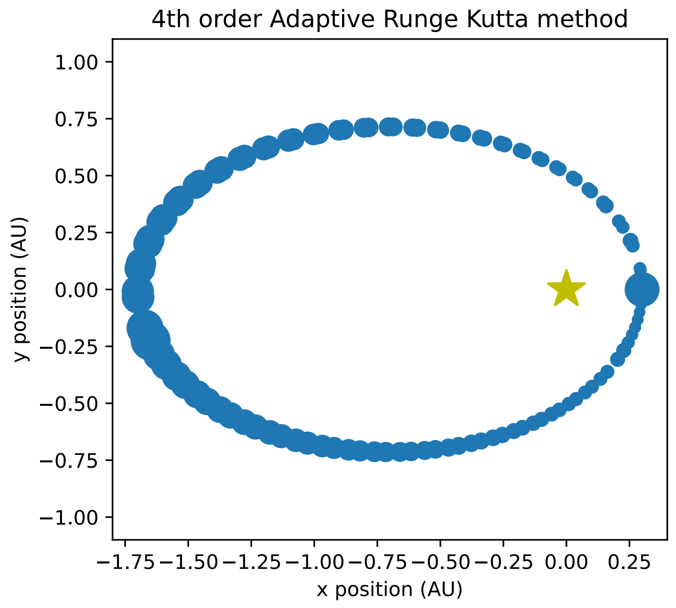
- Requires at least 3 RK steps for every two actually used, but usually results in an overall speedup for a given accuracy
- Usually limit how much $\Delta t'$ can differ from Δt (e.g., by less than a factor of two) in case the denominator happens to diverge

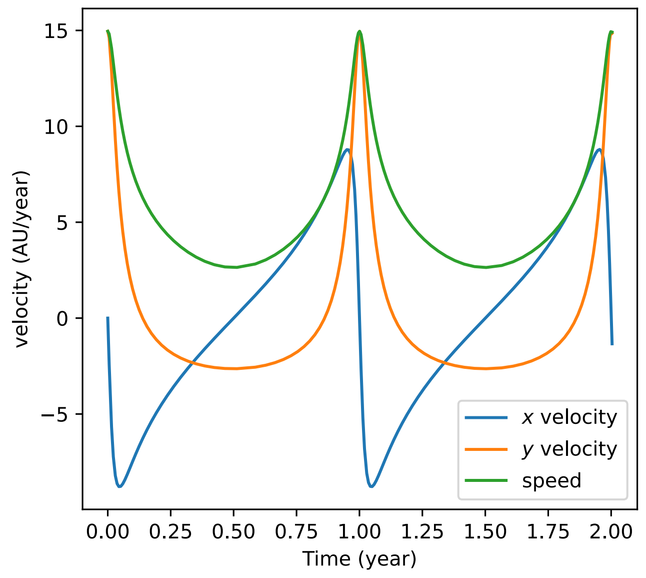
Example: Elliptical orbit with adaptive 4th-order RK

<u>Circular</u>:
x_0= 1 AU
v_{y0} =6.283185 AU/year









<u>Elliptical</u>:
x_0= 0.3 AU
v_{y0} =14.955378 AU/year

Improving the results with local extrapolation

- We can use our knowledge of the error to improve our estimate for *y*(*t*+Δ*t*) recall that:

$$y(t + 2\Delta t) = y_{\Delta t} + 2c\Delta t^5$$

- And:

$$\epsilon = c\Delta t^5 = \frac{1}{30}(y_{\Delta t} - y_{2\Delta t})$$

- So:

$$y(t + 2\Delta t) = y_{\Delta t} + \frac{1}{15}(y_{\Delta t} - y_{2\Delta t}) + \mathcal{O}(\Delta t^6)$$

- No estimate of the error but presumably better than previous 4th order result

Today's lecture:
More on ordinary differential equations

- Orbit example for Euler method
- Runge-Kutta and adaptive RK method
- Beyond RK: Other methods for ODEs
    - Leapfrog/Verlet/modified midpoint
    - Bulirsch-Stoer Method
- Boundary Value problems
- Eigenvalue problems

Leapfrog method

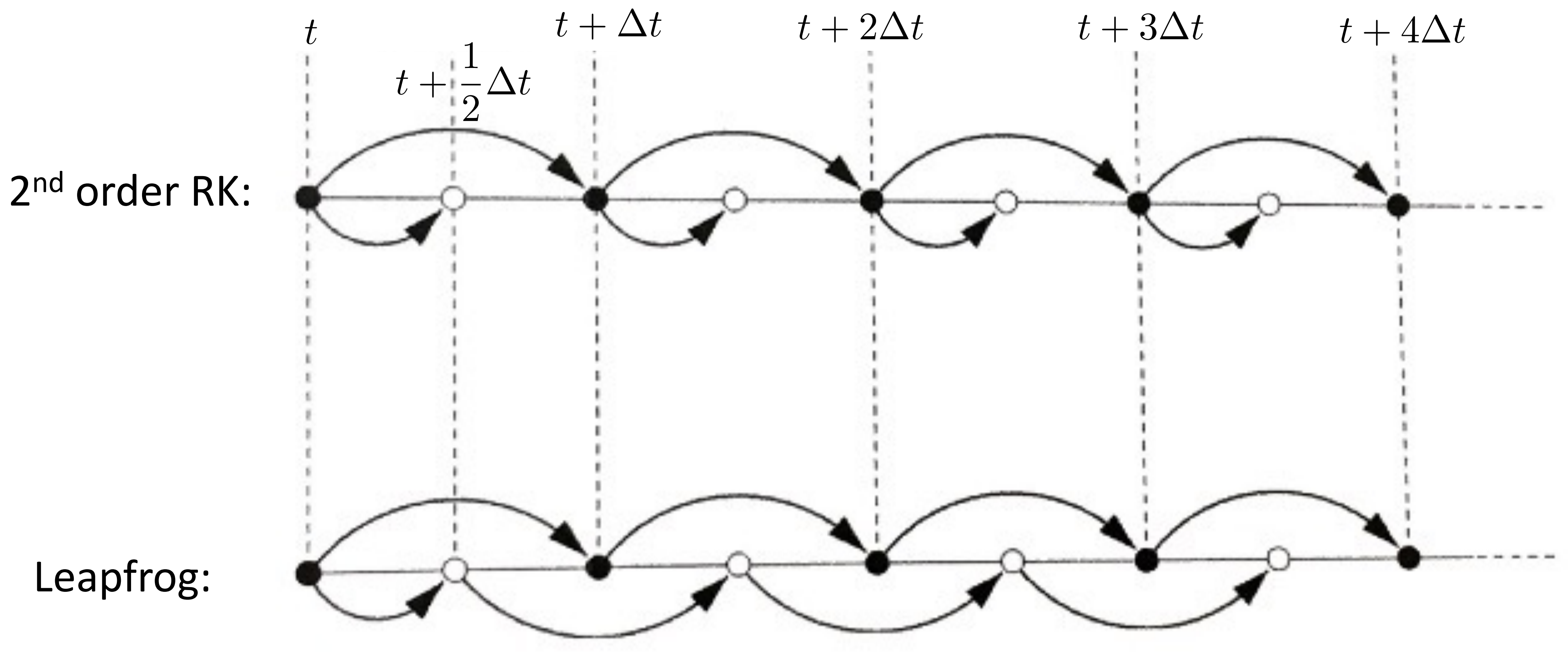
- Recall the second-order RK method:
  - Using the Euler method applied to *t* to estimate the value of a variable at the midpoint of the interval *t* + 1/2Δ*t*

$$y(t+\frac{1}{2}\Delta t) = y(t) + \frac{1}{2}\Delta t f(y,t)$$

$$y(t+\Delta t) = y(t) + \Delta t f\left[y(t+\frac{1}{2}\Delta t), t+\frac{1}{2}\Delta t\right]$$

- Leapfrog method uses a similar approach, except calculates the next midpoint by using the Euler method evaluated at the previous midpoint

Leapfrog method versus 2nd order RK



(Newman)

Leapfrog method

- Starts out the same as RK:

$$y(t + \frac{1}{2}\Delta t) = y(t) + \frac{1}{2}\Delta t f(y, t)$$

$$y(t + \Delta t) = y(t) + \Delta t f\left[y(t + \frac{1}{2}\Delta t), t + \frac{1}{2}\Delta t\right]$$

- Then:

$$y(t + \frac{3}{2}\Delta t) = y(t + \frac{1}{2}\Delta t) + \Delta t f\left[y(t + \Delta t), t + \Delta t\right]$$

$$y(t + 2\Delta t) = y(t + \Delta t) + \Delta t f\left[y(t + \frac{3}{2}\Delta t), t + \frac{3}{2}\Delta t\right]$$

Why the leapfrog method?

- Time reversal symmetric
  - Useful for physics problems where energy conservation is important

- Error is even in step size
  - Ideal starting point for Richardson extrapolation for Bulirsch-Stoer

Leapfrog method is "time-reversal symmetric"

- If we use –Δt instead of Δt, we should retrace our steps
- To see this, start with the equations we repeatedly apply for the Leapfrog method:

$$y(t+\Delta t) = y(t) + \Delta t f\left[y(t+\frac{1}{2}\Delta t), t+\frac{1}{2}\Delta t\right]$$

$$y(t+\frac{3}{2}\Delta t) = y(t+\frac{1}{2}\Delta t) + \Delta t f\left[y(t+\Delta t), t+\Delta t\right]$$

- Set step size to –Δt :

$$y(t-\Delta t) = y(t) - \Delta t f\left[y(t-\frac{1}{2}\Delta t), t-\frac{1}{2}\Delta t\right]$$

$$y(t-\frac{3}{2}\Delta t) = y(t-\frac{1}{2}\Delta t) - \Delta t f\left[y(t-\Delta t), t-\Delta t\right]$$

Leapfrog method is "time-reversal symmetric"

- Now make a trivial shift in time: $t \to t + \frac{3}{2}\Delta t$
- To get:

$$y(t + \frac{1}{2}\Delta t) = y(t + \frac{3}{2}\Delta t) - \Delta t f\left[y(t + \Delta t), t + \Delta t\right]$$

$$y(t) = y(t + \Delta t) - \Delta t f\left[y(t + \frac{1}{2}\Delta t), t + \frac{1}{2}\Delta t\right]$$

- Same as the original: (but moving backwards)

$$y(t + \Delta t) = y(t) + \Delta t f\left[y(t + \frac{1}{2}\Delta t), t + \frac{1}{2}\Delta t\right]$$

$$y(t + \frac{3}{2}\Delta t) = y(t + \frac{1}{2}\Delta t) + \Delta t f\left[y(t + \Delta t), t + \Delta t\right]$$

What about 2nd order Runge-Kutta?

- Original expressions:

$$y(t+\frac{1}{2}\Delta t) = y(t) + \frac{1}{2}\Delta t f(y,t)$$

$$y(t+\Delta t) = y(t) + \Delta t f\left[y(t+\frac{1}{2}\Delta t), t+\frac{1}{2}\Delta t\right]$$

- Set step size to $-\Delta t$:

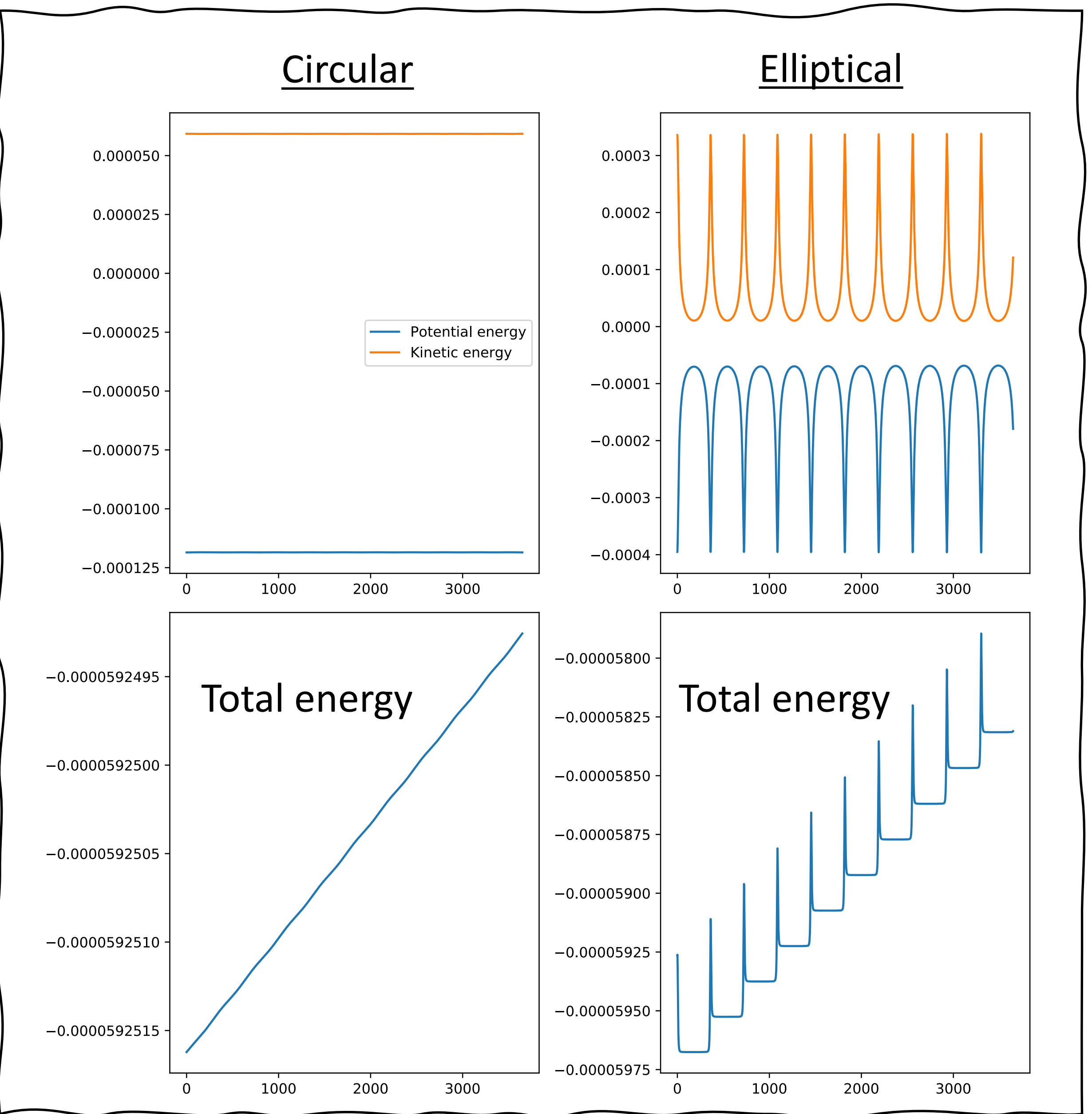
$$y(t-\frac{1}{2}\Delta t) = y(t) - \frac{1}{2}\Delta t f(y,t)$$

$$y(t-\Delta t) = y(t) - \Delta t f\left[y(t-\frac{1}{2}\Delta t), t-\frac{1}{2}\Delta t\right]$$

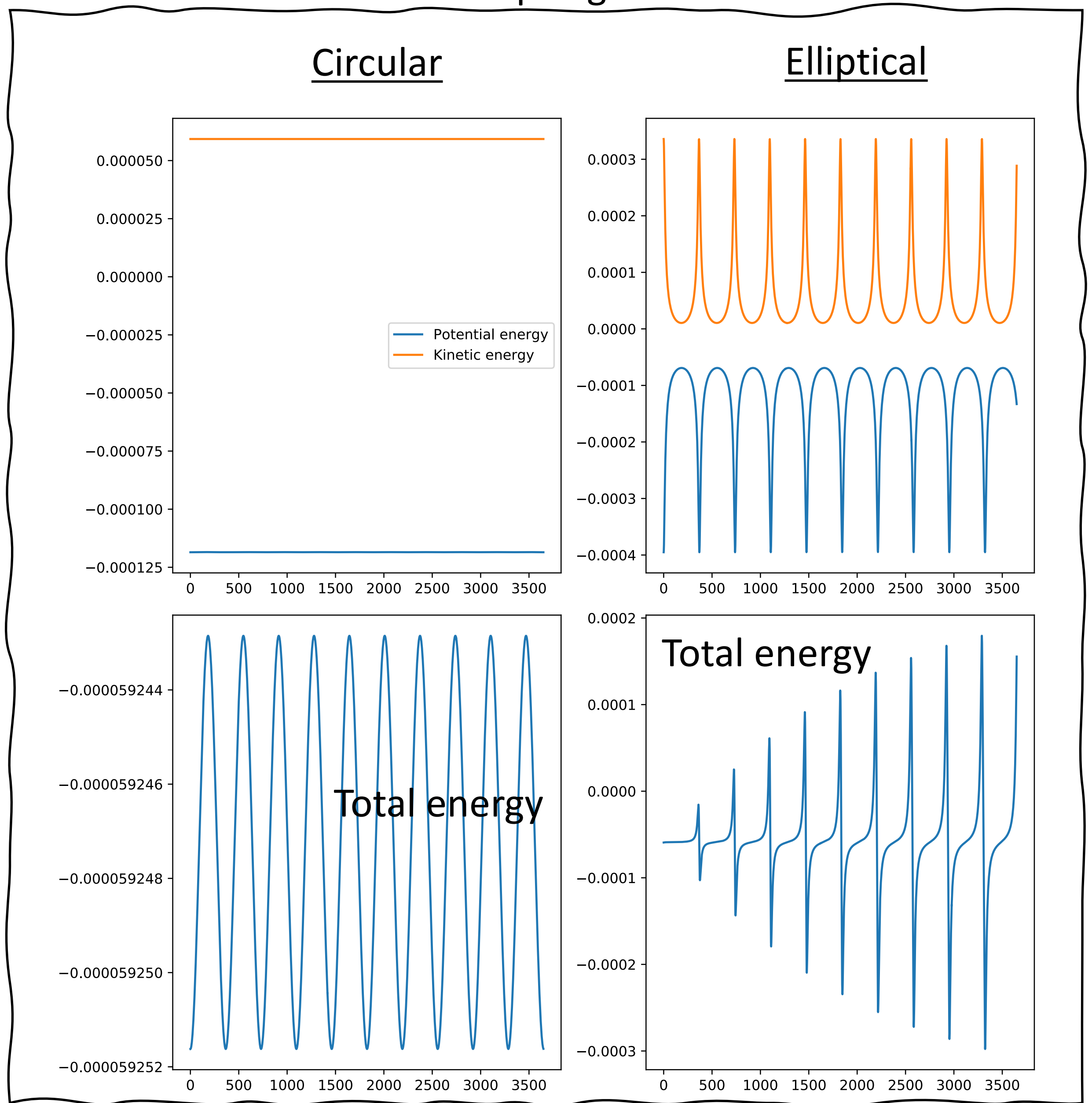
- No way to, e.g., make a shift in *t* to get back to original operations in the opposite direction
    - Errors will result in broken time-reversal symmetry

Why is time-reversal symmetry important? Energy conservation!

2nd order RK



Leapfrog



Verlet method for equations of motion using leapfrog method

- For this method we will limit ourselves to ODEs of the form of equations of motion:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(t), \qquad \frac{d\mathbf{v}}{dt} = \mathbf{f}(\mathbf{x}, t)$$

- (i.e., where the RHS of the first equation does not depend on **x**)
- In that case, we can do the leapfrog method with two equations

Position only at integer steps

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}\left(t + \frac{1}{2}\Delta t\right)$$

Velocity only at half-integer steps

$$\mathbf{v}(t + \frac{3}{2}\Delta t) = \mathbf{v}(t + \frac{1}{2}\Delta t) + \Delta t \mathbf{f}\left[\mathbf{x}(t+\Delta t), t + \Delta t\right]$$

What if we want to know, e.g., the total energy at a point?

- Total energy requires knowing **x** and **v** at the same point
- Let's just step the velocity back half a step with Euler's method:

$$\mathbf{v}(t+\frac{1}{2}\Delta t) = \mathbf{v}(t+\Delta t) - \frac{1}{2}\Delta t\mathbf{f}\left[\mathbf{x}(t+\Delta t), t+\Delta t\right]$$

- Rearrange to get:

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t+\frac{1}{2}\Delta t) + \frac{1}{2}\Delta t\mathbf{f}\left[\mathbf{x}(t+\Delta t), t+\Delta t\right]$$

- Gives velocity at integer points from quantities we have already calculated

Verlet method: Leapfrog in this specific situation of, e.g., EOM:

- First do an initial half step:

$$\mathbf{v}(t+\frac{1}{2}\Delta t) = \mathbf{v}(t) + \frac{1}{2}\Delta t \mathbf{f}\left[\mathbf{x}(t), t\right]$$

- Then repeatedly apply:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \Delta t \mathbf{v}\left(t+\frac{1}{2}\Delta t\right)$$

$$\mathbf{k} = \Delta t \mathbf{f}\left[\mathbf{x}(t+\Delta t), t+\Delta t\right]$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t+\frac{1}{2}\Delta t) + \frac{1}{2}\mathbf{k}$$

$$\mathbf{v}(t+\frac{3}{2}\Delta t) = \mathbf{v}(t+\frac{1}{2}\Delta t) + \mathbf{k}$$

Error of leapfrog/Verlet is even in step size

- Error for a single step is proportional to Δt^3 to leading order
- What about the other orders? Time reversal symmetry gives:

$$\epsilon(-\Delta t) = -\epsilon(\Delta t)$$

- So, the error is an odd function:

$$\epsilon(\Delta t) = c_3 \Delta t^3 + c_5 \Delta t^5 + c_7 \Delta t^7 + ...$$

- But total error is one order less when we accumulate over all steps:

$$\epsilon_{\text{tot}}(\Delta t) = \epsilon(\Delta t) \times \frac{t_f - t_0}{\Delta t}$$

- So:

$$\epsilon_{\text{tot}}(\Delta t) = b_2 \Delta t^2 + b_4 \Delta t^4 + b_6 \Delta t^6 + ...$$

Wait, what about initial Euler half step?

$$\mathbf{v}(t + \frac{1}{2}\Delta t) = \mathbf{v}(t) + \frac{1}{2}\Delta t \mathbf{f}\left[\mathbf{x}(t), t\right]$$

- Introduces odd (and even) higher-order errors
- We can get rid of these errors with the following procedure.

Removing errors from initial Euler half step

- Define variable at integer and half steps:

$$x_0^{\text{int}} = x(t)$$

$$x_1^{\text{half}} = x_0^{\text{int}} + \frac{1}{2}\Delta t f(x_0^{\text{int}}, t)$$

- Then:

$$\begin{aligned}
x_1^{\text{int}} &= x_0^{\text{int}} + \Delta t f(x_1^{\text{half}}, t + \frac{1}{2}\Delta t) \\
x_2^{\text{half}} &= x_1^{\text{half}}, t + \Delta t f(x_1^{\text{int}}, t + \Delta t) \\
x_2^{\text{int}} &= x_1^{\text{int}} + \Delta t f(x_2^{\text{half}}, t + \frac{3}{2}\Delta t) \\
\vdots & \qquad\qquad \vdots \\
x_{m+1}^{\text{half}} &= x_m^{\text{half}} + \Delta t f(x_m^{\text{int}}, t + m\Delta t) \\
x_{m+1}^{\text{int}} &= x_m^{\text{int}} + \Delta t f(x_{m+1}^{\text{half}}, t + (m + \frac{1}{2})\Delta t)
\end{aligned}$$

Removing errors from initial Euler half step: Modified midpoint method

- Take t_f as the final time of the calculation, achieved at step *n*
- We can write the final solution for $x(t+t_f)$ in two ways:

$$x(t+t_f) = x_n^{\text{int}} = x_n^{\text{half}} + \frac{1}{2}\Delta t f(x_n^{\text{int}}, t+t_f)$$

- Or we can use the average of the two:

$$x(t+t_f) = \frac{1}{2}\left[x_n^{\text{int}} + x_n^{\text{half}} + \frac{1}{2}\Delta t f(x_n^{\text{int}}, t+t_f)\right]$$

- This cancels the error from the initial Euler step!
  - Proved by mathematician William Gragg in 1965
- Modified midpoint method: Using the iterative steps from the previous slide and the above expression for $x(t+t_f)$

Bulirsch-Stoer Method

- Why do we care about the modified midpoint method and even-powered errors? They are the basis of the Bulirsch-Stoer Method

- This method combines the modified midpoint method with Richardson extrapolation (e.g., the Romberg method for integrals)

Simple example of Bulirsch-Stoer: First order ODE with one variable

- Equation: $\frac{dx}{dt} = f(x,t)$
- We would like to solve from t to t_f, with $x(t)$ given

- Start by using the modified midpoint method with a single step $\Delta t_1 = t_f$
  - More specifically, two half steps
  - Call this estimate $R_{1,1}$

- Now perform the calculation for $\Delta t_2 = 1/2\ t_f$ to get $R_{2,1}$

Performing Richardson extrapolation

- We can write the "exact" expressions since we know the form of the errors (using $\Delta t_1 = 2\Delta t_2$)

$$x(t+t_f) = R_{2,1} + c_1 \Delta t_2^2 + \mathcal{O}(\Delta t_2^4)$$

$$x(t+t_f) = R_{1,1} + c_1 \Delta t_1^2 + \mathcal{O}(\Delta t_1^4) = R_{1,1} + 4c_1 \Delta t_2^2 + \mathcal{O}(\Delta t_2^4)$$

- So:

$$c_1 \Delta t_2^2 = \frac{1}{3}(R_{2,1} - R_{1,1})$$

- And:

New estimate accurate to fourth order!

$$x(t+t_f) = \underbrace{R_{2,1} + \frac{1}{3}(R_{2,1} - R_{1,1})}_{R_{2,2}} + \mathcal{O}(\Delta t_2^4)$$

Performing Richardson extrapolation, cont.

- Let's do another step: Calculate $R_{3,1}$ with Δt_3=1/3 t_f
- Following the same steps as before:

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

- Then we can write the "exact" result:

$$x(t + t_f) = R_{3,2} + c_2 \Delta t_3^4 + \mathcal{O}(\Delta t_3^6)$$

- From what we had previously:

$$x(t + t_f) = R_{2,2} + c_2 \Delta t_2^4 + \mathcal{O}(\Delta t_2^6) = R_{2,2} + \frac{81}{16} c_2 \Delta t_3^4 + \mathcal{O}(\Delta t_3^6)$$

- Equating these gives: $c_2 \Delta t_3^4 = \dfrac{16}{65}(R_{3,2} - R_{2,2})$

Performing Richardson extrapolation, cont.

- So, we have: $$x(t + t_f) = \underbrace{R_{3,2} + \frac{16}{65}(R_{3,2} - R_{2,2})}_{R_{3,3}} + \mathcal{O}(\Delta t_3^6)$$

  $\uparrow$ New estimate accurate to sixth order!

- Where: $$R_{3,3} = R_{3,2} + \frac{16}{65}(R_{3,2} - R_{2,2})$$

- Three modified midpoint steps, and already have a sixth-order error
  - Gain two orders of accuracy with each step

General Richardson extrapolation

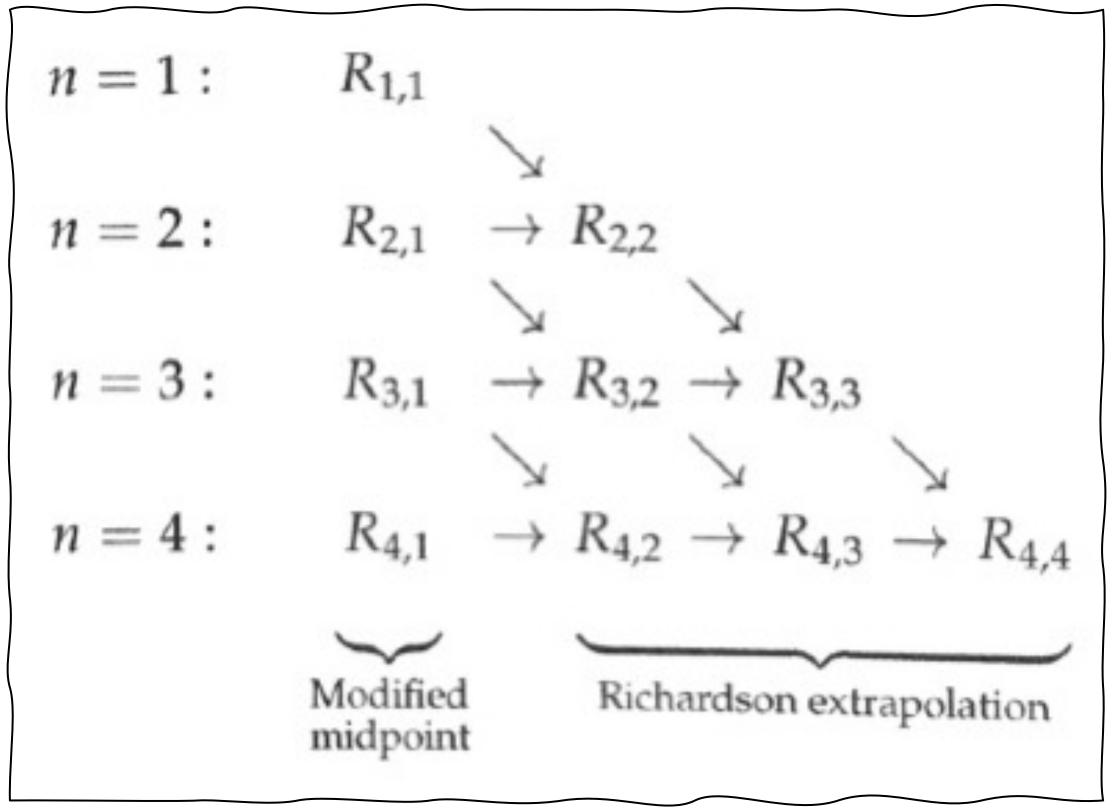
- *n* is the number of modified midpoint steps, which gives us $R_{n,1}$
  - Can obtain $R_{n,m}$ for $m < n$

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

  - See Newman Sec. 8.5

- Which gives an estimate of the result:

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



$$\begin{tikzcd}
n=1: & R_{1,1} \arrow[dr] & & & \\
n=2: & R_{2,1} \arrow[r] \arrow[dr] & R_{2,2} \arrow[dr] & & \\
n=3: & R_{3,1} \arrow[r] \arrow[dr] & R_{3,2} \arrow[r] \arrow[dr] & R_{3,3} \arrow[dr] & \\
n=4: & R_{4,1} \arrow[r] & R_{4,2} \arrow[r] & R_{4,3} \arrow[r] & R_{4,4}
\end{tikzcd}$$

(Newman)

Comments about Bulirsch-Stoer

- Adaptive method: Provides error and estimate
  - Continue until error is below a given accuracy
- Similar approach to Romberg integration with some key differences
  - Increase number of intervals by one in BS instead of doubling in Romberg
  - Not possible to "reuse" previous points like in Romberg
- Only provides accurate estimate for final result $x(t+\mathrm{t}_f)$
  - At intermediate points, we just get raw midpoint method estimates (accurate to Δt^2)
  - Not well suited if we need many (100's or 1000's) steps, so only for rather small regions, where we can get accuracy with < 8 steps
- Can divide larger intervals into smaller ones and apply the BS method
- Often gives better accuracy with less work then RK, especially for relatively smooth functions
  - RK should be used for ODEs with pathological behavior, large fluctuations, divergences, etc.

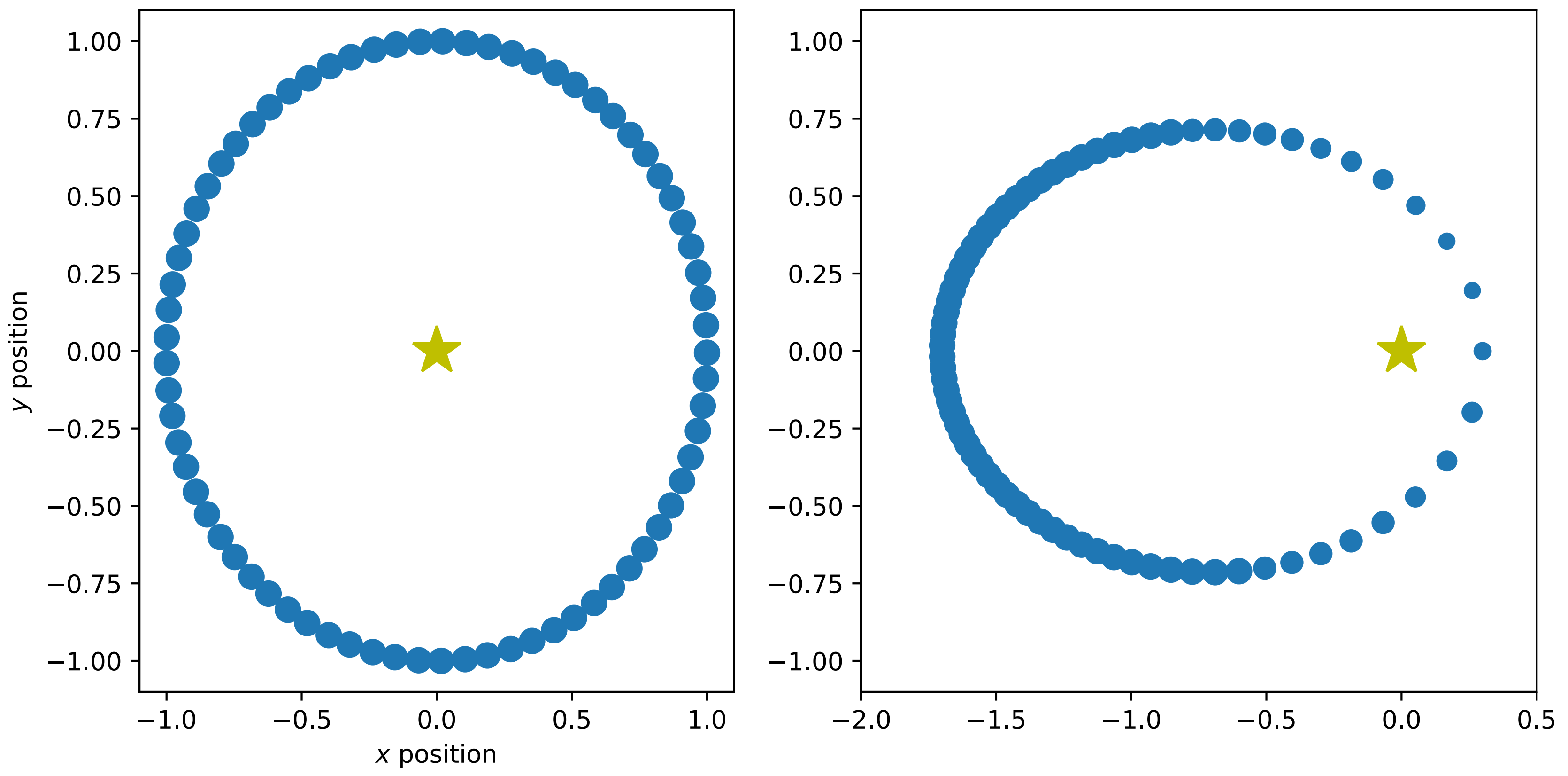
Bulirsch-Stoer Method: Summary

- Say we would like to solve an ODE from t to t_f up to accuracy δ per step
- First, divide the total range into N equal intervals of length t_H. Then do the following steps for each interval:

- 1. Perform a modified midpoint step with one interval from t to t_H to get $R_{1,1}$
- 2. Increase the number of intervals by one to n and calculate $R_{\mathrm{n},1}$ with the modified midpoint method
- 3. Calculate the "row" via Richardson extrapolation, i.e., $R_{n,2}$…$R_{n,n}$
- 4. Compare the error to the target accuracy δt_H. If it is larger than the target accuracy, return to step 2. If it is less than the target accuracy, go to the next interval.

D

Example: Orbits with the Bulirsch-Stoer method



After class tasks

- **NO CLASS/OFFICE HOURS THURSDAY Sept. 28**

- Homework 2 due Oct. 5 by 11pm
  - Office hours: Mondays, 3:00pm to 4:00pm; Thursdays, 10:00am to 1:00pm
    - Feel free to send me an email, and remember, if you push your changes, I should be able to see them
- Readings:
  - Newman Ch. 8