Ordinary Differential Equations

Massimo Ricotti

ricotti@astro.umd.edu

University of Maryland

- *NRiC* §16.
- ODEs involve derivatives with respect to *one* independent variable, e.g., time t.
- ODEs can always be reduced to a *set* of first-order equations (i.e., involving *only* first derivatives). E.g.,

$$\frac{d^2y}{dt^2} + b(t)\frac{dy}{dt} = c(t)$$

is equivalent to the set

$$\begin{aligned} \frac{dy}{dt} &= z(t), \\ \frac{dz}{dt} &= c(t) - b(t)z(t). \end{aligned}$$

- - Example: gravity! In 1-D,

    $$F = ma = m\ddot{x} = -\frac{GMm}{x^2} = F_g.$$

    Let $v(t) = \dot{x}$. Then $\dot{v} = -GM/x^2$. In 3-D, just write out equations for each component (we'll see this again...).
- Usually new variables just derivatives of old, but sometimes need additional factors of t to avoid pathologies.
- General problem is solving set of 1$^{\mathrm{st}}$-order ODEs,

$$\frac{dy_i}{dt} = f_i'(t, y_1, ..., y_N),$$

- where the f'_i are known functions.[a]
- But, also need boundary conditions: algebraic conditions on values of y_i at discrete time(s) t...

[a] Often ODEs are coupled to begin with, e.g., classic Lotka-Volterra predator-prey model:

$$\begin{aligned} \dot{x} &= Ax - Bxy - ex, \\ \dot{y} &= -Cy + Dxy - dy. \end{aligned}$$

Here x and y might represent the population of rabbits and foxes, respectively. Then A is the reproduction rate of the rabbits, B is the consumption rate of rabbits by the foxes, C is the death rate by natural causes of the foxes, and D is the population increase rate of the foxes due to consumption of rabbits. We've also added terms with coefficients d and e representing the hunting rate by humans. For $d = e = 0$, the equilibrium solution of this system is cyclical.

Another Astrophysical Example:

Time-dependent chemistry of the ISM/IGM

For pure Hydrogen gas:

$$\begin{aligned} \dot{n}_{HI} &= -\Gamma n_{HI} - C n_{HI} n_e + \alpha n_p n_e, \\ n_e &= n_p, \\ n &= n_p + n_{HI} = const \end{aligned}$$

Adding Helium:

$$\begin{aligned} \dot{n}_{HeI} &= -\Gamma_1 n_{HeI} - C_1 n_{HeI} n_e + \alpha_1 n_{HeII} n_e, \\ \dot{n}_{HeIII} &= \Gamma_2 n_{HeII} + C_2 n_{HeII} n_e - \alpha_2 n_{HeIII} n_e, \\ n_e &= n_p + n_{HeII} + 2 n_{HeIII}, \\ n &= n_{HeI} + n_{HeII} + n_{HeIII} = const \end{aligned}$$

ODE Boundary Conditions (BCs)

- Two categories of BC:
  1. Initial Value Problem (IVP): all y_i's are given at some starting point t_s, and solution is needed from t_s to t_f.
  2. Two-point Boundary Value Problem (BVP): y_i are specified at two or more t, e.g., some at t_s, some at t_f (only one BC needed for each y_i).
- Generally, IVP much easier to solve than 2-pt BVP, so consider this first.

Finite Differences

- How do you represent derivatives with a discrete number system?
- Basic idea: replace dy/dt with finite differences $\Delta y/\Delta t$. Then:

$$\lim_{\Delta t \to 0} \frac{\Delta y}{\Delta t} \to \frac{dy}{dt}.$$

- How do you use this to solve ODEs?

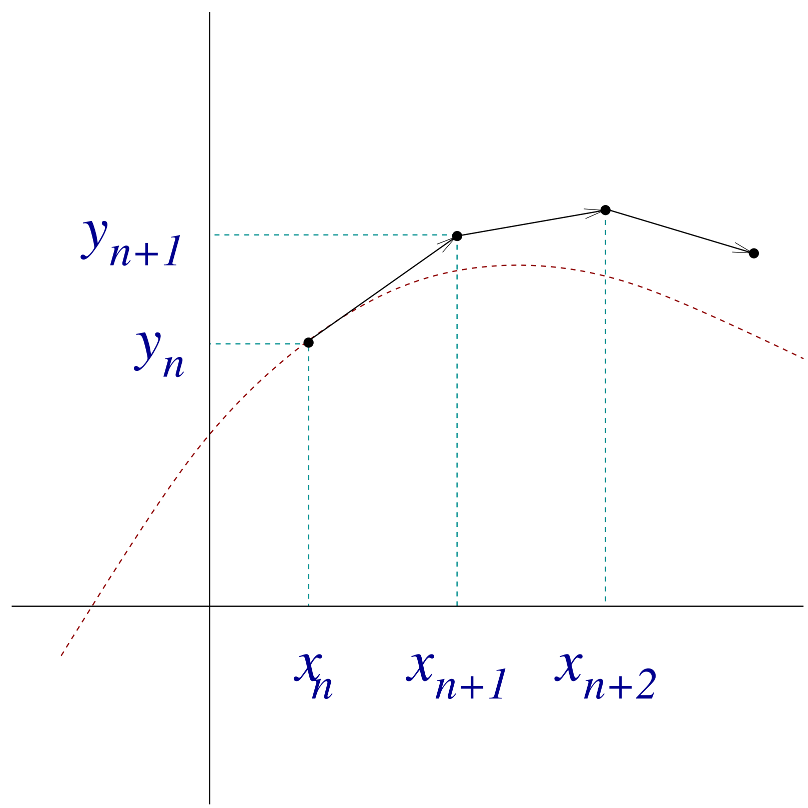
Euler's Method

- Write $\Delta \mathbf{y}/\Delta t = \mathbf{f}'(t, \mathbf{y}) \Rightarrow \Delta \mathbf{y} = \Delta t \, \mathbf{f}'(t, \mathbf{y})$.
- Start with known values $\mathbf{y}_n$ at t_n (initial values).
- Then $\mathbf{y}_{n+1}$ at $t_{n+1} = t_n + h$ is

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}'(t_n, \mathbf{y}_n).$$

- h is called the *step size*.

- Integration is not symmetric: derivative evaluated only at start of step $\Rightarrow$ error term $\mathcal{O}(h^2)$, from Taylor series ($f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + ...$). So, Euler's method is first order.



- Example: consider $\dot{y} = y$ with $y(0) = 1$. We know the solution to be $y = e^t$. Using Euler's method with $h = 1/2$, we find

$$
\begin{aligned}
y_0 &= 1, \\
y_1 &= y_0 + y_0/2 = 3/2, \\
y_2 &= y_1 + y_1/2 = 9/4, \\
y_3 &= y_2 + y_2/2 = 27/8, \\
&\vdots \\
y_n &= \left(\tfrac{3}{2}\right)^n,
\end{aligned}
$$

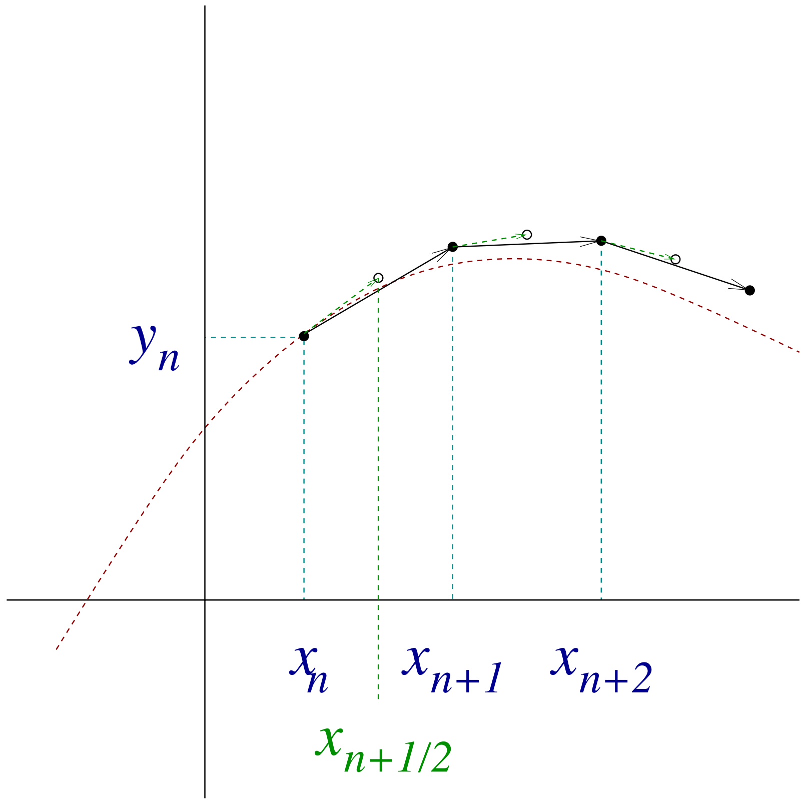
i.e., the solution is always $\leq e^t$ (since $t = nh = n/2$ and $e^{1/2} \doteq 1.65$).

Runge-Kutta Methods

- We can do better by symmetrizing the derivative:
  - Take a trial Euler step to midpoint: compute $t_{n+1/2}$ and evaluate $\mathbf{y}_{n+1/2}$.
  - Use these to evaluate derivative $\mathbf{f}'(t_{n+1/2}, \mathbf{y}_{n+1/2})$.
  - Then use this to go back and take a full step.
- Thus:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}'\left[t_n + \frac{1}{2}h, \mathbf{y}_n + \frac{1}{2}h\mathbf{f}'(t_n, \mathbf{y}_n)\right] + \mathcal{O}(h^3).$$

- Can show that $\mathcal{O}(h^2)$ terms "cancel," so leading error term is $\mathcal{O}(h^3)$, giving 2$^{\rm nd}$-order Runge-Kutta (midpoint method).



- Following previous example, first step using midpoint method:

$$\begin{aligned} y_1 &= y_0 + (1/2)f'(0 + 1/4, 1 + (1/4)f'(0, 1)), \\ &= 1 + (1/2)f'(1/4, 5/4), \\ &= 1 + (1/2)(5/4), \\ &= 1 + 5/8, \\ &= 1.625. \end{aligned}$$

- The idea behind midpoint method is to use Euler but with derivative at midpoint:

$$y(t+h) = y(t) + hf'(t + \frac{1}{2}h) = y(t) + h\left[f'(t) + \frac{1}{2}hf''(t)\right] + \mathcal{O}(h^3).$$

  This is essentially a Taylor series <u>within</u> a Taylor series.

- Use Euler to <u>determine</u> derivative at midpoint:

$$\begin{aligned} k_1 &= hf'(t_n, y_n), \\ k_2 &= hf'(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1), \\ y_{n+1} &= y_n + k_2 + \mathcal{O}(h^3). \end{aligned}$$

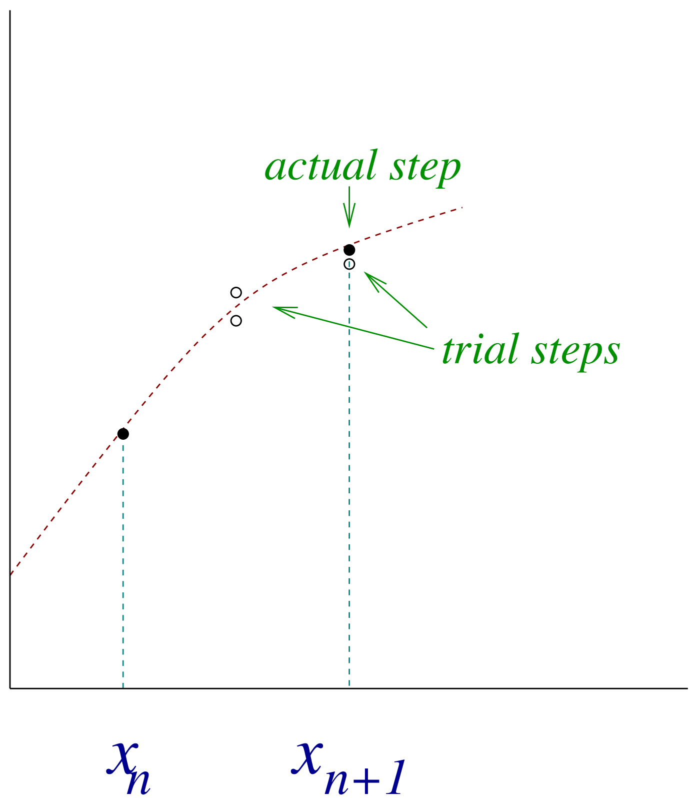
Fourth-order Runge-Kutta

- Actually, there are many ways to evaluate $\mathbf{f}'$ at midpoints, which add higher-order error terms with different coefficients. Can add these together in ways such that higher-order error terms cancel. E.g., can build 4$^{\text{th}}$-order Runge-Kutta (RK4):

$$\begin{aligned} \mathbf{k}_1 &= h\mathbf{f}'(t_n, \mathbf{y}_n), \\ \mathbf{k}_2 &= h\mathbf{f}'(t_n + h/2, \mathbf{y}_n + \mathbf{k}_1/2), \\ \mathbf{k}_3 &= h\mathbf{f}'(t_n + h/2, \mathbf{y}_n + \mathbf{k}_2/2), \\ \mathbf{k}_4 &= h\mathbf{f}'(t_n + h, \mathbf{y}_n + \mathbf{k}_3). \end{aligned}$$

Then:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{k}_1/6 + \mathbf{k}_2/3 + \mathbf{k}_3/3 + \mathbf{k}_4/6 + \mathcal{O}(h^5).$$


actual step
trial steps
x_n
x_{n+1}

```
#include "nrutil.h"

void rk4(float y[], float dydx[], int n, float x, float h, float yout[],
void (*derivs)(float, float [], float []))

int i;
float xh,hh,h6,*dym,*dyt,*yt;

dym=vector(1,n);
dyt=vector(1,n);
yt=vector(1,n);
hh=h*0.5;
h6=h/6.0;
xh=x+hh;
for (i=1;i<=n;i++) yt[i]=y[i]+hh*dydx[i];
(*derivs)(xh,yt,dyt);
for (i=1;i<=n;i++) yt[i]=y[i]+hh*dyt[i];
(*derivs)(xh,yt,dym);
for (i=1;i<=n;i++)
yt[i]=y[i]+h*dym[i];
dym[i] += dyt[i];

(*derivs)(x+h,yt,dyt);
for (i=1;i<=n;i++)
yout[i]=y[i]+h6*(dydx[i]+dyt[i]+2.0*dym[i]);
free_vector(yt,1,n);
free_vector(dyt,1,n);
free_vector(dym,1,n);

/* (C) Copr. 1986-92 Numerical Recipes Software ?421.1-9. */
```

- Disadvantage of RK4: requires $\mathbf{f}'$ to be evaluated 4 times per step.
- But, can still be cost effective if larger steps OK.
- RK4 is a workhorse method. Higher-order RK4 takes too much effort for increased accuracy.
- Other methods (e.g., Bulirsch-Stoer, *NRiC* §16.4) are more accurate for smooth functions.
- But RK4 often “good enough.”