Solving Classical Equations of Motion

Newton's equation of motion govern the dynamics of:

- solar systems, galaxies…
  (some relativistic effects, some times large, e.g., binary start systems)
- "everyday motion"; projectiles, baseballs, mechanical machines
- non-electronic aspects of solids, liquids and gases
  (some quantum effects for light atoms/molecules)
  - potentials from quantum mechanics

We will discuss basic numerical algorithms
(discretized time axis)

- 1D equation, properties of different integrators
- 2D motion, driving forces, disspation…
- chaos in classical dynamic systems (will be postponed to later)

1D motion - single particle x(t)

velocity and acceleration:

$$\dot{x}(t) = \frac{dx(t)}{dt} = v(t), \quad \ddot{x}(t) = \frac{d^2x(t)}{dt^2} = a(t)$$

Given a force F, the equation of motion is

$$\ddot{x}(t) = \frac{1}{m} F[x(t), \dot{x}(t), t]$$

- potential depending on x or x(t)
- dissipation (friction) depending on v
- location independent driving (t dep)

This 2nd order diff equation can be written as 2 coupled 1st order eqs:

$$\dot{x}(t) = v(t)$$
$$\dot{v}(t) = a[x(t), v(t), t]$$

Discretized time axis (constant time step):

$$t \in \{t_0, t_1, \ldots, t_N\}, \quad \Delta_t = t_{i+1} - t_i$$

Start with given initial conditions at t_0: $x=x_0$, $v=v_0$

- use approximation to diff equation to integrate forward in time

To go from time t_n to t_{n+1}, in general we can expand around t_n

$$x_{n+1} = x_n + \Delta_t v_n + \tfrac{1}{2}\Delta_t^2 a_n + \tfrac{1}{6}\Delta_t^3 \dot{a}_n + \ldots$$

$$v_{n+1} = v_n + \Delta_t a_n + \tfrac{1}{2}\Delta_t^2 \dot{a}_n + \tfrac{1}{6}\Delta_t^3 \ddot{a}_n + \ldots$$

Simplest case: go to linear order:

Euler forward algorithm

$$x_{n+1} = x_n + \Delta_t v_n$$

$$v_{n+1} = v_n + \Delta_t a_n$$

error in x_{n+1} is $O(\Delta_t^2)$

Julia implementation

```
for i=1:nt
    t=dt*(i-1)
    dowhatever(x,v,t)
    a=acc(x,v,t)
    x=x+dt*v
    v=v+dt*a
end
```

the path [x(t),v(t)] may be written to a file or processed in some other way.

Note: for consistency with the series expansion, the acceleration should be evaluated with “old” x,v,t (i-1)

The Euler algorithm is not very good in practice
- should not be used in any serious work

Illustration of the Euler method

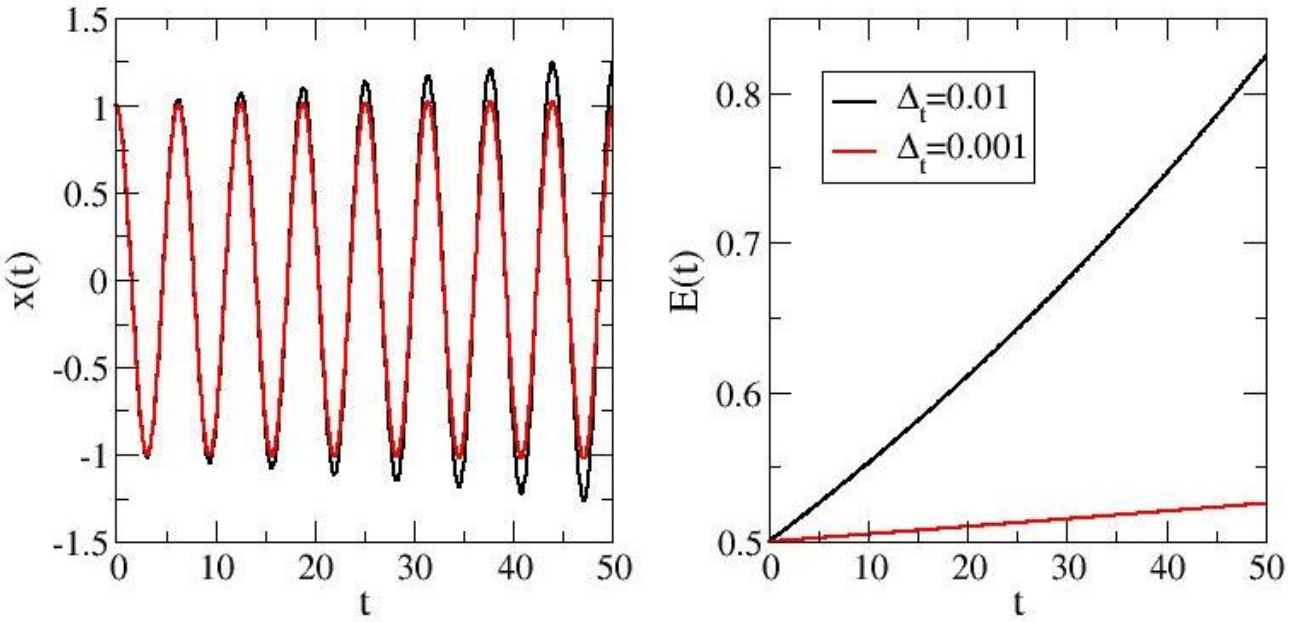
Harmonic oscillator

$$E = \frac{1}{2}kx^2 + \frac{1}{2}mv^2 \quad (F = -kx)$$

Periodic motion; $\omega = \sqrt{\frac{k}{m}}$

Test case: k=m=1, x_0=1, v_0=0

Comparing two different time steps: Δ_t=0.01, 0.001



Amplitude increases with time, unbounded energy error

- there are algorithms with bounded energy error for periodic motion

Leapfrog (Verlet) Algorithm

Use second-order form of x:

$$x_{n+1} = x_n + \Delta_t v_n + \tfrac{1}{2}\Delta_t^2 a_n + O(\Delta_t^3)$$

and recall first-order form of v:

$$v_{n+1} = v_n + \Delta_t a_n + O(\Delta_t^2)$$

Re-write x formula as

$$x_{n+1} = x_n + \Delta_t(v_n + \tfrac{1}{2}\Delta_t a_n) + O(\Delta_t^3)$$

where we can identify "half-step" velocity $v_{n+1/2} = v_n + \tfrac{1}{2}\Delta_t a_n + O(\Delta_t^2)$

$$x_{n+1} = x_n + \Delta_t v_{n+1/2} + O(\Delta_t^3)$$

We will only have v on the half-step, use $v_{n-1/2}$ to obtain $v_{n+1/2}$
(cubic step error remains intact)

$$v_{n+1/2} = v_{n-1/2} + \Delta_t a_n$$

$$x_{n+1} = x_n + \Delta_t v_{n+1/2}$$

$$v_{n+1/2} = v_n + (\Delta_t/2)a_n + (\Delta_t/2)^2 \dot{a}_n + O(\Delta_t^3)$$

$$v_{n-1/2} = v_n - (\Delta_t/2)a_n + (\Delta_t/2)^2 \dot{a}_n - O(\Delta_t^3)$$

$$\rightarrow v_{n+1/2} = v_{n-1/2} + a_n \Delta_t + O(\Delta_t^3)$$

Note: The acceleration (force) at the integer step n is required

- **we do not have v_n**
- **force cannot be v-dependent here; $a_n = a_n(x_n, t)$**