

# How Can We Do Better Than Euler Algorithm?

- 1.) Extrapolate in step size
- 2.) Better approximation for  $v(t)$ .

Recall:

$$\frac{dx}{dt} = v(t), \quad \text{so}$$

$$x(\Delta) = x(0) + \int_0^{\Delta} dt v(t) \quad \text{exact}$$

For Euler, we approximate

$$v(t) = v(0) \quad \text{velocity at beginning of interval.}$$

For Euler-Cromer,

$$v(t) = v(\Delta) \quad \text{velocity at end of interval}$$

In each case, ~~the~~  $v(t)$  is treated as a constant. So at each step, we make an error of order  $\Delta^2$

$$\text{i.e. } v(t) = v(0) + t v'(0) + \frac{1}{2} t^2 v''(0) \quad \text{Taylor series}$$

We neglect

$$\int_0^{\Delta} t v'(0) dt = \frac{1}{2} \Delta^2 v'(0) + \text{higher order terms}$$

## Mid point method

Taylor expand  $v(t)$  around  $\Delta/2$ .

$$v(t) = v(\Delta/2) + (t - \Delta/2) v'(\Delta/2) + \frac{1}{2} (t - \Delta/2)^2 v''(\Delta/2)$$

$$\int_0^\Delta dt v(t) = \Delta v(\Delta/2) + 0 \cdot v'(\Delta/2) + O(\Delta^3)$$

By this clever artifice, we eliminate

the order  $\Delta^2$  error and only have  $\Delta^3$  local error.

But, this required  $v$  at middle of interval.

Mid point method uses average velocity to approximate mid point velocity.

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

(5.35<sup>7</sup>)

$$x_{n+1} = x_n + \frac{1}{2} (v_n + v_{n+1}) \Delta$$

(5.36<sup>7</sup>)

$$\text{or } x_{n+1} = x_n + v_n \Delta + \frac{1}{2} a_n \Delta^2$$

5.38

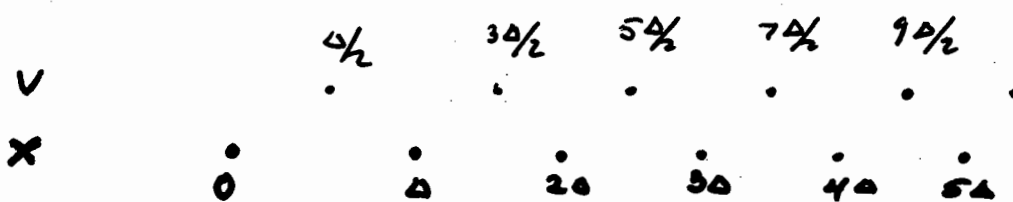
This is more accurate for  $x$  than for  $v$ .

### Leap Frog or Half step method

We saw above, if we know  $v(t/2)$ , we have more accuracy in updating  $x$ .

Same would be true in updating  $v$ , need  $x$  halfway between steps in  $v$ .

### Leap frog



$$X_{n+1} = X_n + V_{n+1/2} \Delta$$

$$V_{n+1/2} = V_{n-1/2} + a_n \Delta$$

Not self starting! But, once started, can continue indefinitely.

## Runge Kutta

We know it is better to use derivative at midpoint of time interval. Use Euler to predict  $y$  at midpoint, then use midpoint derivative to take full step.

$$\frac{dy}{dx} = f(x, y)$$

$$\text{Euler } y_{n+1} = y_n + \Delta f(x_n, y_n)$$

$$\text{Let } k_1 = \Delta f(x_n, y_n)$$

$$k_2 = \Delta f(x_n + \frac{\Delta}{2}, y_n + \frac{1}{2}k_1)$$



estimate of  $y$  at mid time

$$y_{n+1} = y_n + k_2 + O(\Delta^3)$$

Note 1) self starting

2) must calculate derivative function twice per step.

$$\frac{dy}{dx} = f(x, y)$$

4<sup>th</sup> order Runge-Kutta

$$K_1 = \Delta f(x_n, y_n)$$

$$K_2 = \Delta f\left(x_n + \frac{\Delta}{2}, y_n + \frac{K_1}{2}\right)$$

$$K_3 = \Delta f\left(x_n + \frac{\Delta}{2}, y_n + \frac{K_2}{2}\right)$$

$$K_4 = \Delta f(x_n + \Delta, y_n + K_3)$$

$$y_{n+1} = y_n + \frac{\Delta}{6} (K_1 + 2K_2 + 2K_3 + K_4)$$

local error is order  $\Delta^4$

## Adams - Bashforth

So far, still treating  $V(t)$  as a constant.

Suppose, we find a better approximation to  $V(t)$ .

How about a linear polynomial?

Extrapolate  $V$  from earlier steps

$$t = -\Delta$$

$$t = 0$$

Interpolating polynomial:

$$y(t_1) = y_1 \quad y(t_2) = y_2$$

$$y(t) = \frac{t-t_1}{t_2-t_1} y_2 + \frac{t-t_2}{t_1-t_2} y_1$$

$$V(t) = \frac{(t+\Delta) V_0}{\Delta} - \frac{t}{\Delta} V_{-\Delta} + O(t^2)$$

$$\int_0^{\Delta} V(t) dt = \frac{3}{2} \Delta V_0 - \frac{1}{2} \Delta V_{-\Delta} + O(\Delta^3)$$

So,

$$x(\Delta) = x_0 + \frac{3}{2} v_0 \Delta - \frac{1}{2} v_{-1} \Delta + O(\Delta^3)$$

Must also improve updating for  $v$ .

$$\frac{dv}{dt} = a(t), \text{ so}$$

$$v(\Delta) = v_0 + \frac{3}{2} a_0 \Delta - \frac{1}{2} a_{-1} \Delta + O(\Delta^3)$$

One problem is that Adams-Bashforth is not self starting.

Must use Taylor series or another method to get started.

Runge-Kutta is self starting.

To start leap frog,

$$v_{1/2} = v_0 + \frac{1}{2} a_0 \Delta.$$

This has error of order  $\Delta^2$ , but it is only for one step. (May also do this at end.)

Intermediate steps have error of order  $\Delta^3$ , but to integrate a distance  $t=1$ , require  $\frac{1}{\Delta}$  steps.

Leap frog is ~~inaccurate~~ <sup>has errors of</sup> order  $\Delta^2$  in global error. Second order method.