# PHY407: Computational Physics
# Fall, 2017

Lecture 6: Ordinary differential equations, Part 1

- Runge-Kutte - accuracy
- Leapfrog & Verlet – energy conservation
- Adaptive Time Stepping - efficiency

# Summary & Status

- ☑ Week 1: Python basics and pseudocoding …
- ☑ Week 2: Programming tips, roundoff error, numerical integration intro.
- ☑ Week 3: Gaussian quadrature, numerical differentiation.
- ☑ Week 4: Solving linear and nonlinear equations, eigensystems.
- ❑ Week 5: Solving ordinary differential equations
- ❑ Week 6: Fourier transforms
- ❑ Week 7: More on ODEs.
- ❑ Week 8: PDEs

# Solving ODEs, Part 1

# The task at hand

- Very often in physics we want to solve systems of ordinary differential equations, subject to initial conditions

One dimension: $\frac{dx}{dt} = f(x,t)$, given $x(t=0) = x_0$

# The task at hand

- Very often in physics we want to solve systems of ordinary differential equations, subject to initial conditions

One dimension: $\dfrac{dx}{dt} = f(x,t)$, given $x(t=0) = x_0$

Multiple dimensions: $\dfrac{dx_i}{dt} = f_i(x_1,\ldots,x_n,t)$, $x_i(t=0) = x_{io}$, $i = 1,\ldots,n$

# The task at hand

- Very often in physics we want to solve systems of ordinary differential equations, subject to initial conditions

One dimension: $\frac{dx}{dt} = f(x,t)$, given $x(t=0) = x_0$

Multiple dimensions: $\frac{dx_i}{dt} = f_i(x_1,\ldots,x_n,t)$, $x_i(t=0) = x_{io}$, $i = 1,\ldots,n$

Higher order: e.g. $\frac{d^3x}{dt^3} = g(x,t) \rightarrow \begin{array}{l} \frac{dx}{dt} = v \\ \frac{dv}{dt} = a \\ \frac{da}{dt} = g \end{array}$

# The task at hand

- Very often in physics we want to solve systems of ordinary differential equations, subject to initial conditions

One dimension: $\dfrac{dx}{dt} = f(x,t)$, given $x(t=0) = x_0$

Multiple dimensions: $\dfrac{dx_i}{dt} = f_i(x_1,\ldots,x_n,t)$, $x_i(t=0) = x_{io}$, $i = 1,\ldots,n$

Higher order: e.g. $\dfrac{d^3x}{dt^3} = g(x,t) \rightarrow$ $\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= a \\ \frac{da}{dt} &= g \end{aligned}$

These equations can be complicated or intractable to solve analytically, but easy to solve on a computer.

We've used Euler's method but know it has weaknesses.

# `odeint`: Python's blackbox solver

- Python has a built in ODE solver called "odeint" located in the scipy.integrate module. (Aside: This module also contains a bunch of integration functions that can do gaussian quadrature, simpson's rule etc.).

- More info on the module can be found here:

  http://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html

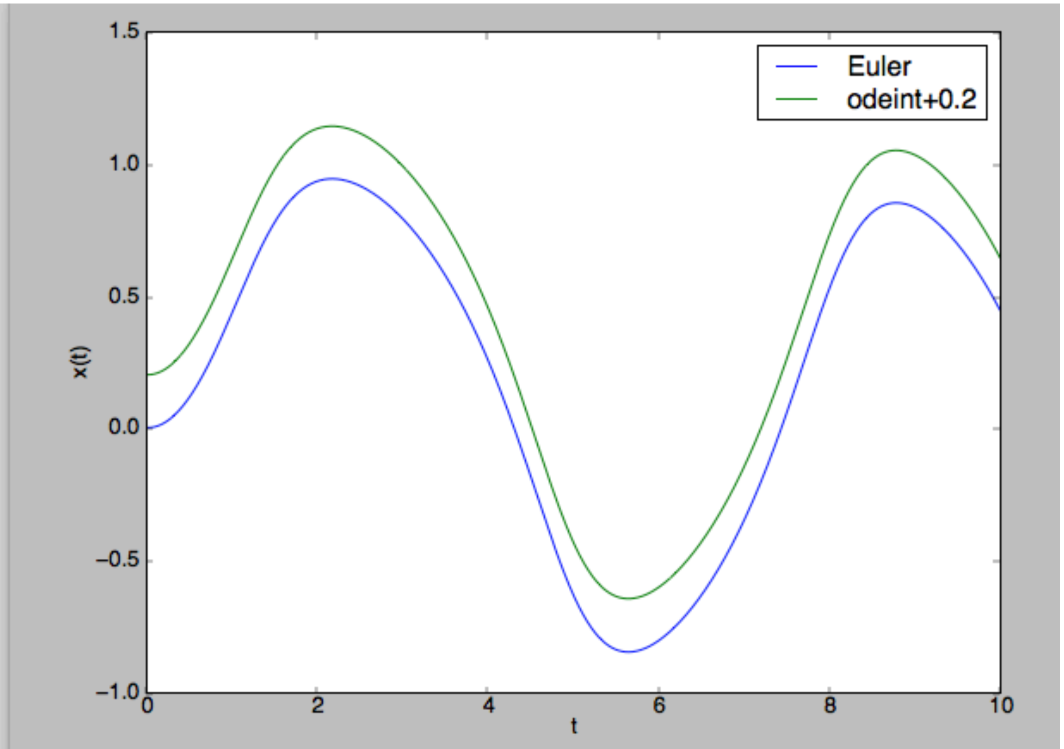
- It really functions as a black box and you don't know how accurate your solution is (since you don't know what method was used).

- If that doesn't matter to your specific application, then just use odeint. However, if it does matter, then you can write your own ODE solver with the method that you want.

```python
a = 0.0              # Start of the interval
b = 10.0             # End of the interval
N = 1000             # Number of steps
h = (b-a)/N          # Size of a single step
x = 0.0              # Initial condition

tpoints = arange(a,b,h)
xpoints = []
for t in tpoints:
    xpoints.append(x)
    x += h*f(x,t)

#also solve by odeint
x_new = odeint(func=f,y0=0,t=tpoints)

plot(tpoints,xpoints)
xlabel("t")
ylabel("x(t)")
```

**Euler**

**odeint**



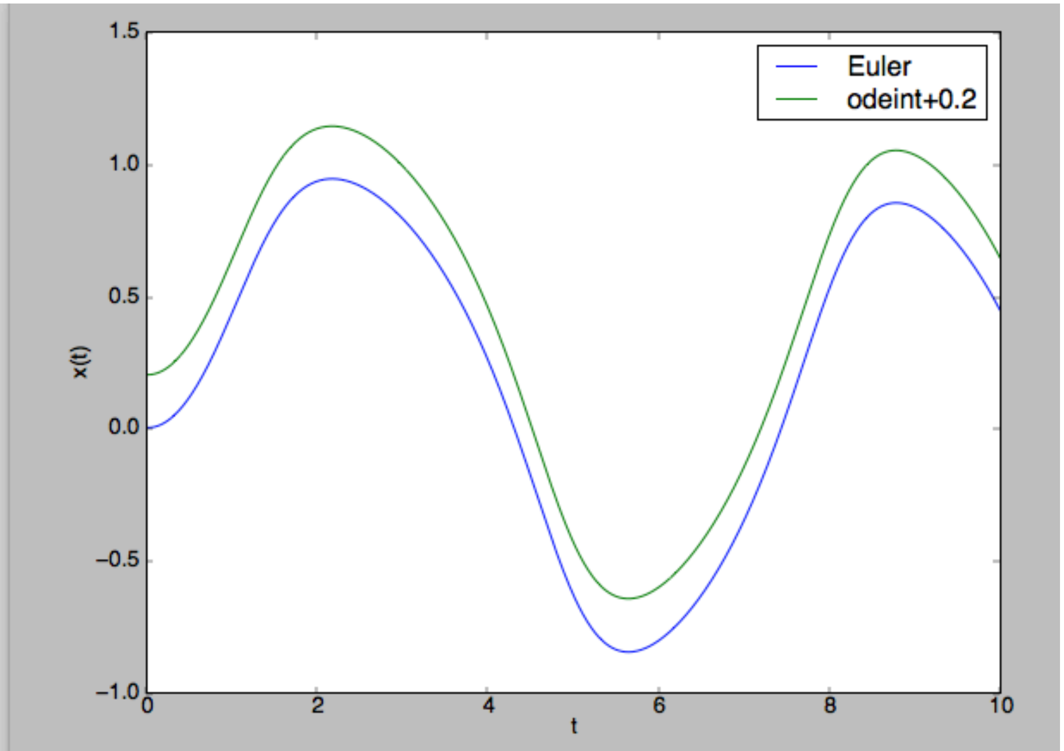

*From euler-odeint.py*

# Moving beyond Euler and Black Boxes

```
a = 0.0                 # Start of the interval
b = 10.0                # End of the interval
N = 1000                # Number of steps
h = (b-a)/N             # Size of a single step
x = 0.0                 # Initial condition

tpoints = arange(a,b,h)
xpoints = []
for t in tpoints:
    xpoints.append(x)
    x += h*f(x,t)

#also solve by odeint
x_new = odeint(func=f,y0=0,t=tpoints)

plot(tpoints,xpoints)
xlabel("t")
ylabel("x(t)")
```

**Euler**

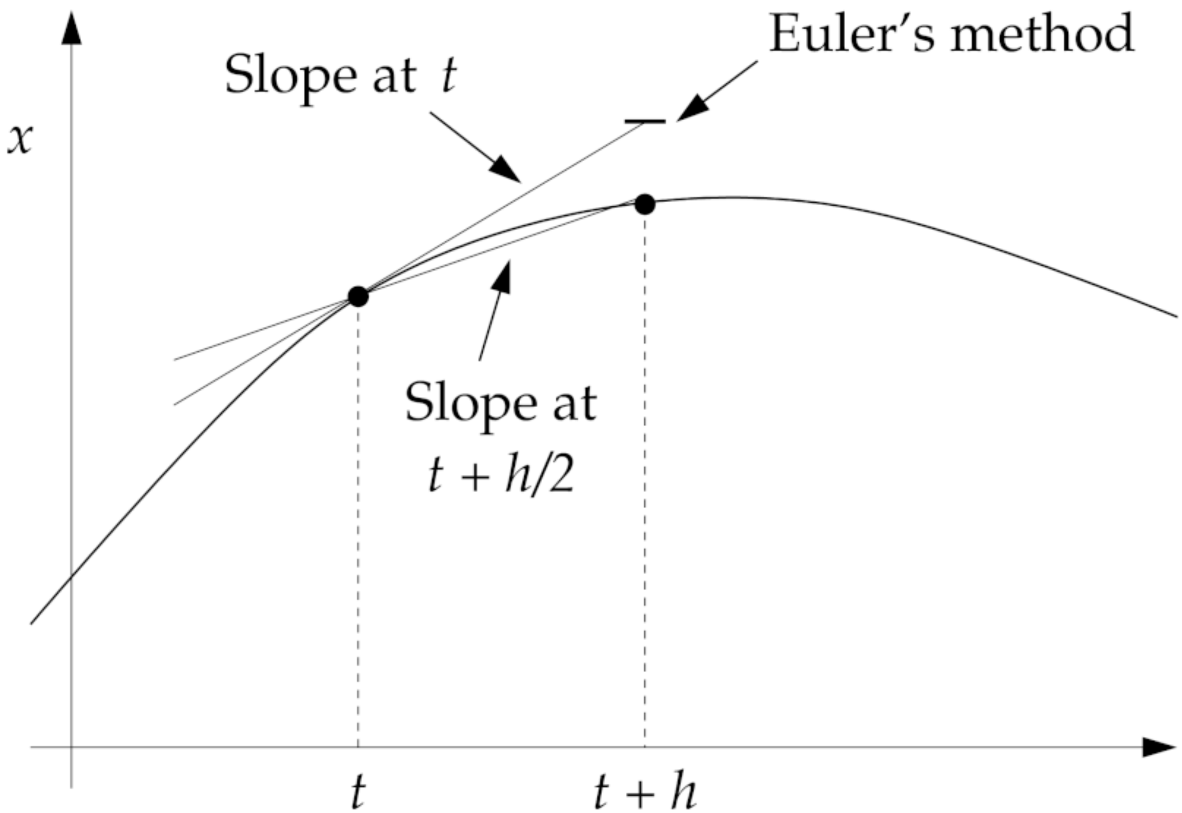
**odeint**



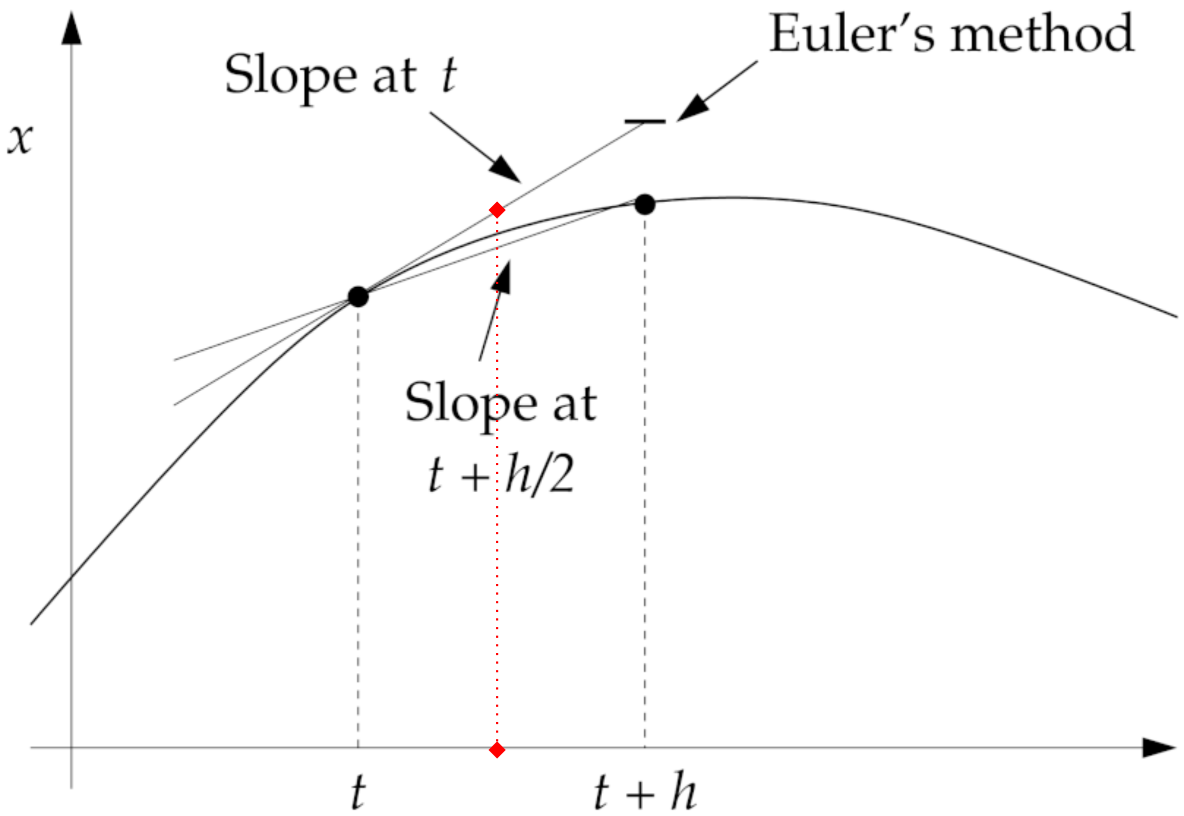

*From euler-odeint.py*

- The Euler method's is $O(h^2)$ accurate at each step, and integrating across the whole interval is $O(h)$.
- There are many choices on how to improve on this.

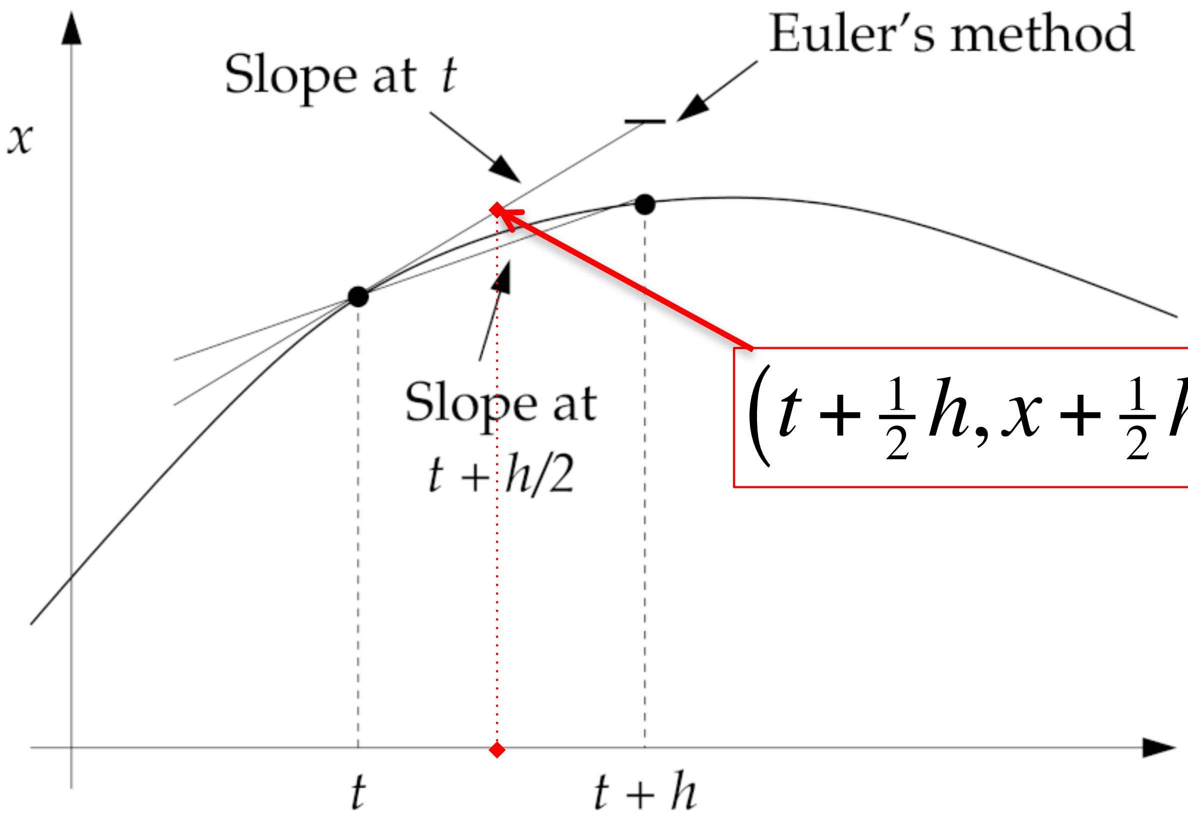
# Runge-Kutte Methods



# Runge-Kutte Methods

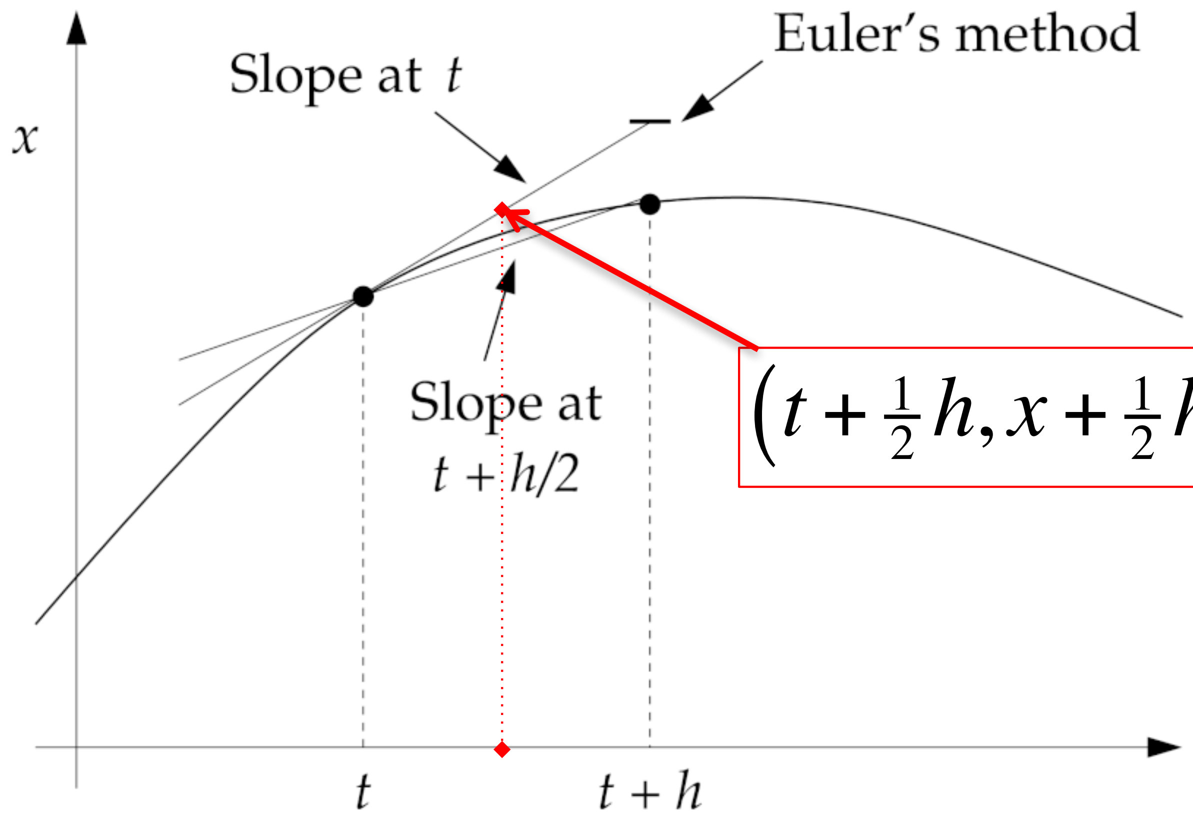


# Runge-Kutte Methods




$$\boxed{\left(t+\tfrac{1}{2}h,\, x+\tfrac{1}{2}hf(x,t)\right)}$$

# Runge-Kutte Methods

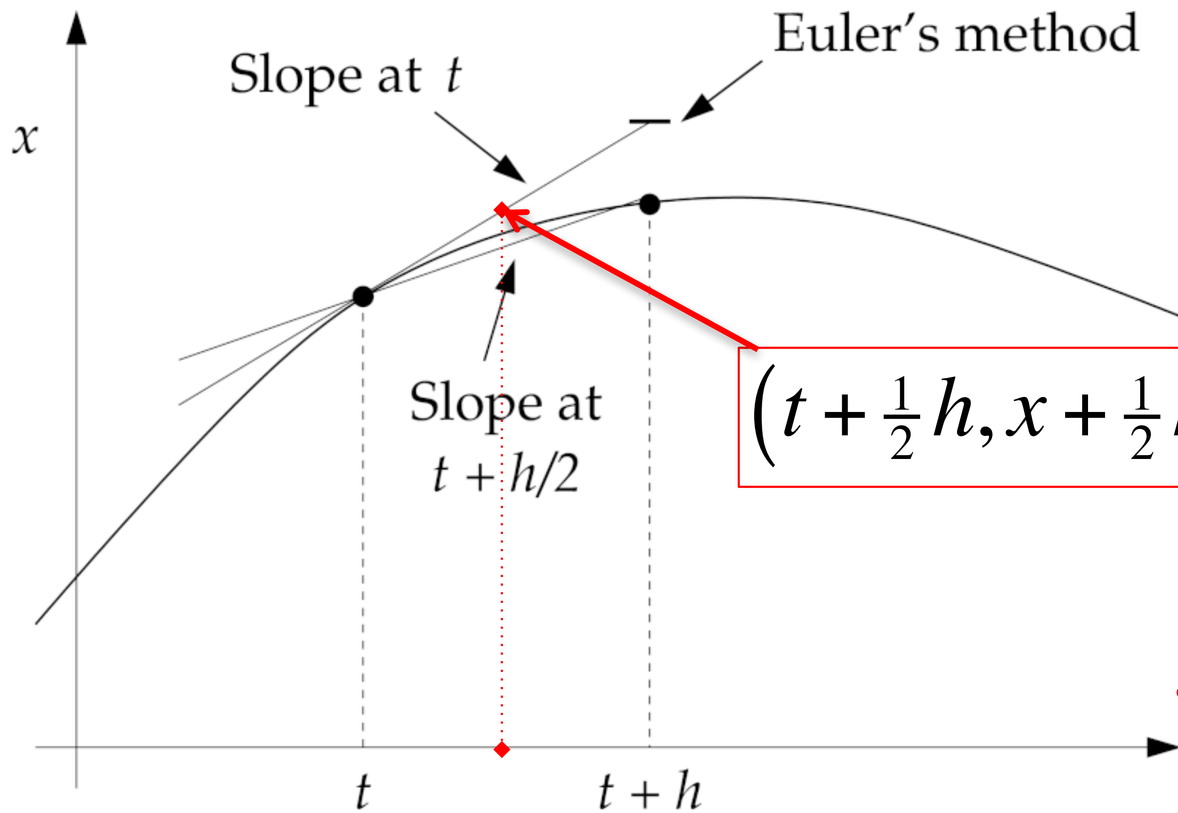



$$\boxed{\left(t+\tfrac{1}{2}h, x+\tfrac{1}{2}hf(x,t)\right)}$$

Second order Runge-Kutte:

$$x(t+h) = x(t) + hf\left(x+\tfrac{1}{2}hf(x,t), t+\tfrac{1}{2}h\right)$$

# Runge-Kutte Methods




$$\left(t + \tfrac{1}{2}h, x + \tfrac{1}{2}hf(x,t)\right)$$

Second order Runge-Kutte:

$$x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t), t + \tfrac{1}{2}h\right)$$

- **Use f (x,t) to estimate the midpoint.**
- **Use midpoint estimate to get improved slope estimate.**
- **Iterative approach leads to O($h^2$) accuracy.**

# Runge-Kutte Methods

```python
from math import sin
from numpy import arange
from pylab import plot,xlabel,ylabel,show

def f(x,t):
    return -x**3 + sin(t)

a = 0.0            # Start of the interval
b = 10.0           # End of the interval
N = 1000           # Number of steps
h = (b-a)/N        # Size of a single step
x = 0.0            # Initial condition

tpoints = arange(a,b,h)
xpoints = []
for t in tpoints:
    xpoints.append(x)
    x += h*f(x,t)

plot(tpoints,xpoints)
xlabel("t")
ylabel("x(t)")
show()
```

```python
from math import sin
from numpy import arange
from pylab import plot,xlabel,ylabel,show

def f(x,t):
    return -x**3 + sin(t)

a = 0.0
b = 10.0
N = 10
h = (b-a)/N

tpoints = arange(a,b,h)
xpoints = []

x = 0.0
for t in tpoints:
    xpoints.append(x)
    k1 = h*f(x,t)
    k2 = h*f(x+0.5*k1,t+0.5*h)
    x += k2

plot(tpoints,xpoints)
xlabel("t")
ylabel("x(t)")
show()
```

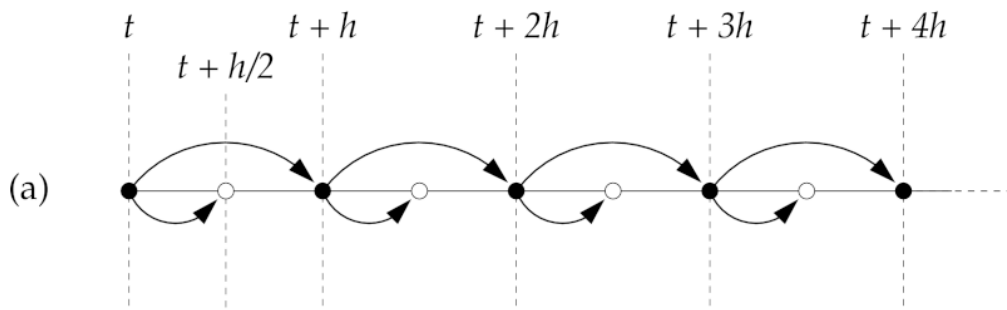
```python
from math import sin
from numpy import arange
from pylab import plot,xlabel,ylabel,show

def f(x,t):
    return -x**3 + sin(t)

a = 0.0
b = 10.0
N = 10
h = (b-a)/N

tpoints = arange(a,b,h)
xpoints = []
x = 0.0

for t in tpoints:
    xpoints.append(x)
    k1 = h*f(x,t)
    k2 = h*f(x+0.5*k1,t+0.5*h)
    k3 = h*f(x+0.5*k2,t+0.5*h)
    k4 = h*f(x+k3,t+h)
    x += (k1+2*k2+2*k3+k4)/6

plot(tpoints,xpoints)
xlabel("t")
ylabel("x(t)")
show()
```

**RK0 (Euler): O(h)** | **RK2: $O(h^2)$** | **RK4: $O(h^4)$**

- To get these formula, use Taylor Expansion about various points to cancel errors at successive orders.

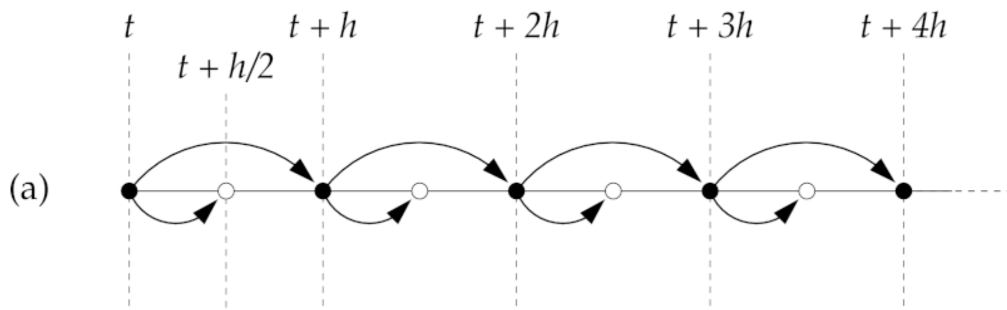
# Leapfrog Methods

$$\boxed{\text{RK2: } x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t),\, t + \tfrac{1}{2}h\right)}$$



# Leapfrog Methods

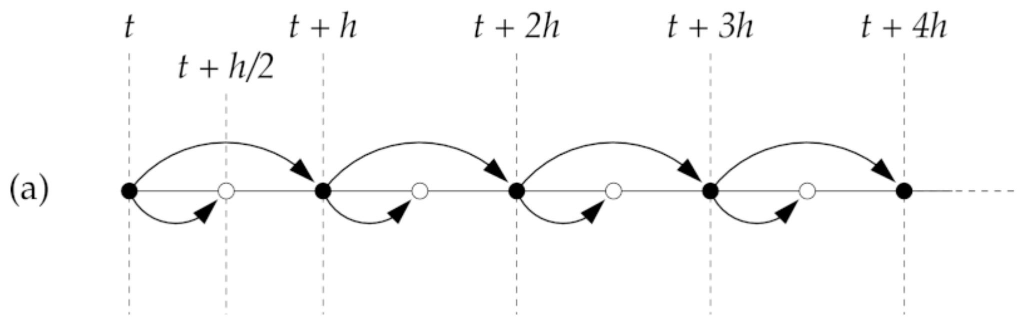
$$\boxed{\text{RK2: } x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t), t + \tfrac{1}{2}h\right)}$$




RK2: Next step requires f info from midpoint ahead.

# Leapfrog Methods

$$\text{RK2: } x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t), t + \tfrac{1}{2}h\right)$$




RK2: Next step requires f info from midpoint ahead.

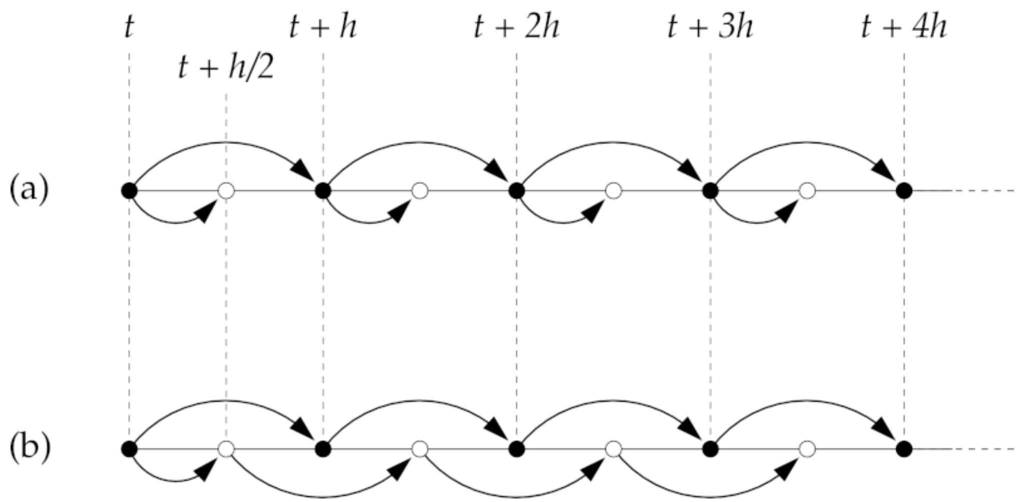
- Leapfrog: next midpoint from last one.

$$x(t+h) = x(t) + hf\left(x\left(t + \tfrac{1}{2}h\right), t + \tfrac{1}{2}h\right)$$

$$x\left(t + \tfrac{3}{2}h\right) = x\left(t + \tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right)$$

# Leapfrog Methods

$$\text{RK2: } x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t), t + \tfrac{1}{2}h\right)$$




RK2: Next step requires f info from midpoint ahead.

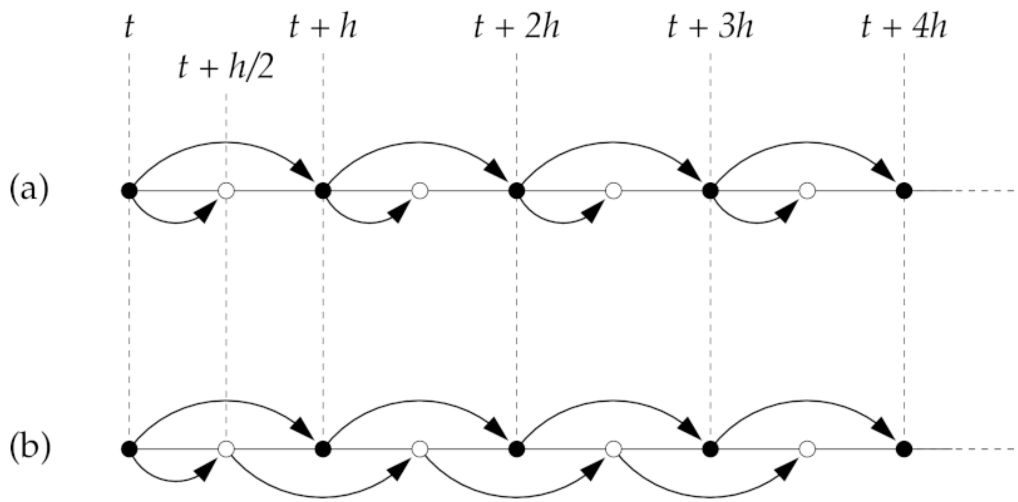
- Leapfrog: next midpoint from last one.

$$x(t+h) = x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right)$$

$$x\left(t+\tfrac{3}{2}h\right) = x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right)$$

# Leapfrog Methods

$$\text{RK2: } x(t+h) = x(t) + hf\left(x + \tfrac{1}{2}hf(x,t), t + \tfrac{1}{2}h\right)$$




RK2: Next step requires f info from midpoint ahead.

LF: Start with RK2 step. Next step requires f info from previous midpoint.

- Leapfrog: next midpoint from last one.

$$x(t+h) = x(t) + hf\left(x\left(t + \tfrac{1}{2}h\right), t + \tfrac{1}{2}h\right)$$

$$x\left(t + \tfrac{3}{2}h\right) = x\left(t + \tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right)$$

# Leapfrog Timestepping is Reversible

$$x(t+h) = x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right)$$

$$x\left(t+\tfrac{3}{2}h\right) = x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right)$$

# Leapfrog Timestepping is Reversible

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right) \end{aligned}}$$

- If we take $h \rightarrow -h,\ t \rightarrow t + 3h/2$

# Leapfrog Timestepping is Reversible

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right) \end{aligned}}$$

- If we take $h \to -h,\ t \to t + 3h/2$

$$\boxed{\begin{aligned} x\left(t+\tfrac{1}{2}h\right) &= x\left(t+\tfrac{3}{2}h\right) - hf\left(x(t+h), t+h\right) \\ x(t) &= x(t+h) - hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \end{aligned}}$$

# Leapfrog Timestepping is Reversible

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right) \end{aligned}}$$

- If we take $h \to -h,\ t \to t + 3h/2$

$$\boxed{\begin{aligned} x\left(t+\tfrac{1}{2}h\right) &= x\left(t+\tfrac{3}{2}h\right) - hf\left(x(t+h), t+h\right) \\ x(t) &= x(t+h) - hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \end{aligned}}$$

- These are the same as the original equations but work backward with identical values.

# Leapfrog Timestepping is Reversible

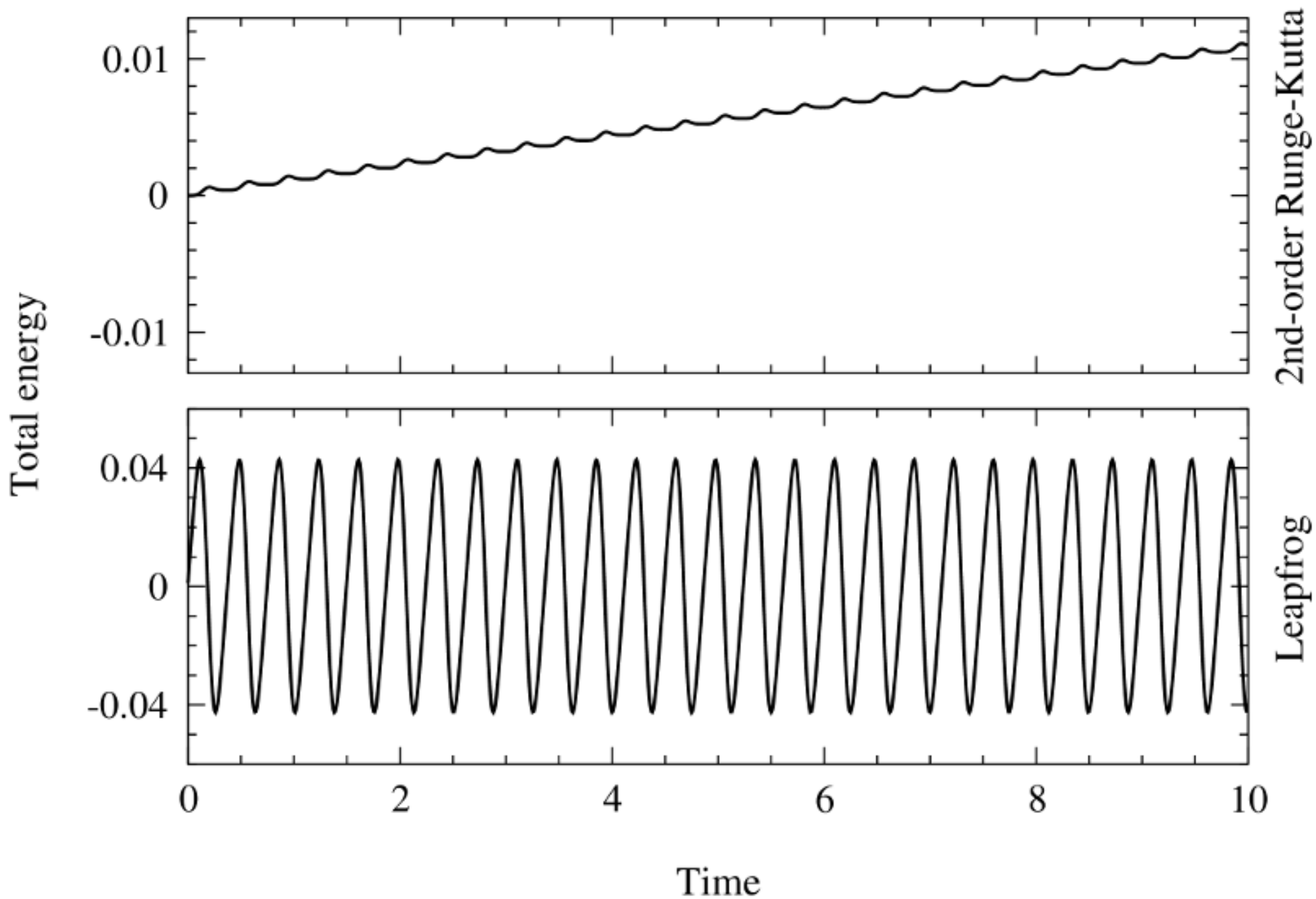
$$\boxed{\begin{aligned} x(t+h) &= x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right) \end{aligned}}$$

- If we take $h \rightarrow -h,\ t \rightarrow t + 3h/2$

$$\boxed{\begin{aligned} x\left(t+\tfrac{1}{2}h\right) &= x\left(t+\tfrac{3}{2}h\right) - hf\left(x(t+h), t+h\right) \\ x(t) &= x(t+h) - hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \end{aligned}}$$

- This indicates that the method is reversible, consistent with energy conservation.

# Energy of a nonlinear pendulum



# Leapfrog to Verlet

$$\boxed{\begin{aligned} x\left(t+h\right) &= x\left(t\right)+hf\left(x\left(t+\tfrac{1}{2}h\right),t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right)+hf\left(x\left(t+h\right),t+h\right) \end{aligned}}$$

# Leapfrog to Verlet

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right) \end{aligned}}$$

- For the special case of Newton's second law, leapfrog leads to the Verlet Method:

# Leapfrog to Verlet

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right) \end{aligned}}$$

- For the special case of Newton's second law, leapfrog leads to the Verlet Method:

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hv\left(t+\tfrac{1}{2}h\right) \\ v\left(t+\tfrac{3}{2}h\right) &= v\left(t+\tfrac{1}{2}h\right) + hF\left(x(t+h), t+h\right)/m \end{aligned}}$$

# Leapfrog to Verlet

$$\boxed{\begin{aligned} x\left(t+h\right) &= x\left(t\right) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x\left(t+h\right), t+h\right) \end{aligned}}$$

- For the special case of Newton's second law, leapfrog leads to the Verlet Method:

$$\boxed{\begin{aligned} x\left(t+h\right) &= x\left(t\right) + hv\left(t+\tfrac{1}{2}h\right) \\ v\left(t+\tfrac{3}{2}h\right) &= v\left(t+\tfrac{1}{2}h\right) + hF\left(x\left(t+h\right), t+h\right)/m \end{aligned}}$$

- This method involves fewer calculations than if we applied leapfrog to each of x and v.

# Verlet is Reversible Too

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right) \end{aligned}}$$

- For the special case of Newton's second law, leapfrog leads to the Verlet Method:

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hv\left(t+\tfrac{1}{2}h\right) \\ v\left(t+\tfrac{3}{2}h\right) &= v\left(t+\tfrac{1}{2}h\right) + hF\left(x(t+h), t+h\right)/m \end{aligned}}$$

- Since Verlet is a leapfrog method, it is also reversible, and in this case its reversibility implies energy conservation.

# Verlet is Reversible Too

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hf\left(x\left(t+\tfrac{1}{2}h\right), t+\tfrac{1}{2}h\right) \\ x\left(t+\tfrac{3}{2}h\right) &= x\left(t+\tfrac{1}{2}h\right) + hf\left(x(t+h), t+h\right) \end{aligned}}$$

- For the special case of Newton's second law, leapfrog leads to the Verlet Method:

$$\boxed{\begin{aligned} x(t+h) &= x(t) + hv\left(t+\tfrac{1}{2}h\right) \\ v\left(t+\tfrac{3}{2}h\right) &= v\left(t+\tfrac{1}{2}h\right) + hF\left(x(t+h), t+h\right)/m \end{aligned}}$$

- Note: In the lab use Equation 8.75c, for v(t+h), to calculate the energy at (t+h).

# Summary

- We have started to explore ordinary differential equation solvers
  - You can use canned routines sometimes.
  - We looked at two midpoint rules: RK2/RK4 and the leapfrog methods.
  - Verlet = Leapfrog for Newton's Second Law.
- You now have enough information to go ahead with Lab06.