



# *Particles - An Introduction to Ordinary Differential Equations ODEs*

---

Simulation in Computer Graphics  
Computer Graphics  
University of Freiburg

WS 09/10

# Outline

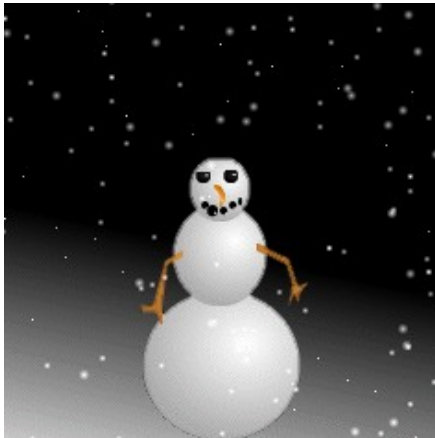


- introduction
- particle motion
- finite differences
- first order ODEs
- second order ODEs

# Motivation



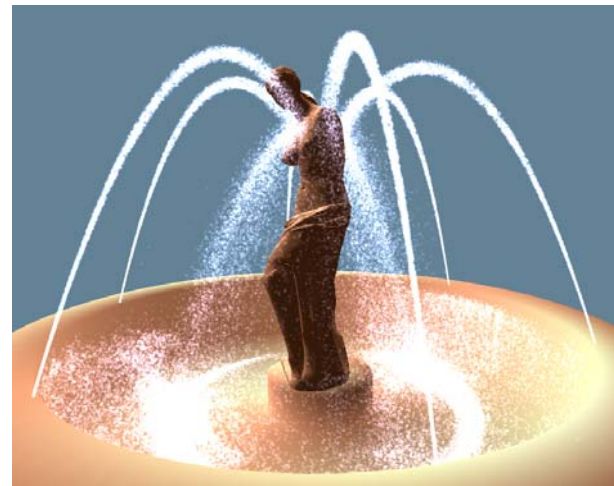
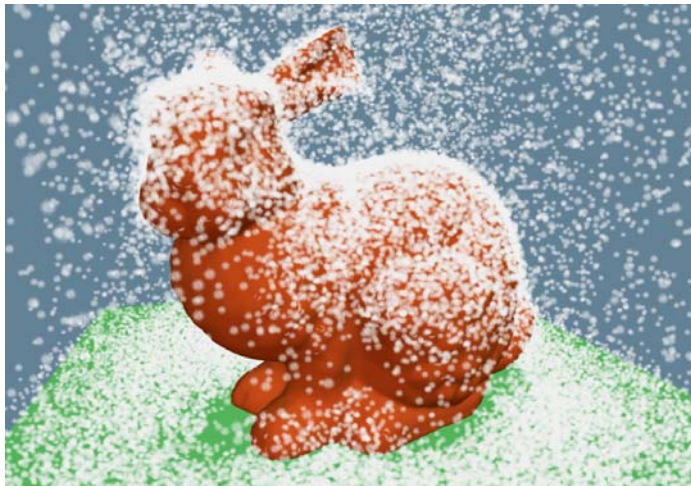
- sets of particles (particle systems) are used to model time-dependent phenomena such as snow, fire, smoke



# Motivation



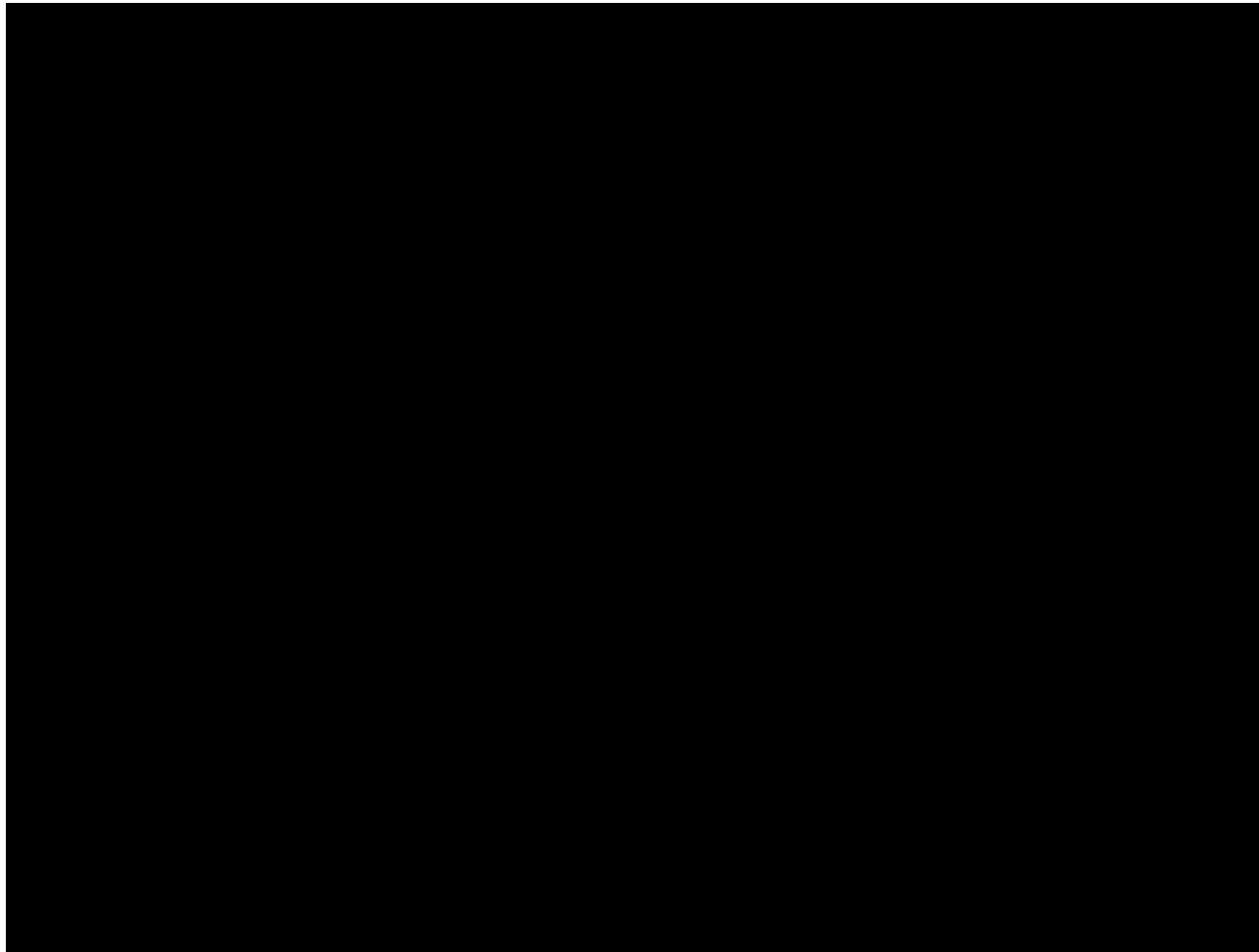
- particles are characterized by mass, position, and velocity
- forces determine the dynamic behavior
- inter-particle forces are neglected
- particles can carry arbitrary attributes for rendering purposes, e. g. shape, color, transparency, life time



Kolb, Latta, Rezk-Salama

University of Freiburg - Computer Science Department - Computer Graphics

# *Demo*



750,000 particles in XNA,  
<http://www.youtube.com/watch?v=CyAZ2Y7nOTw>

University of Freiburg - Computer Science Department - Computer Graphics

# Outline

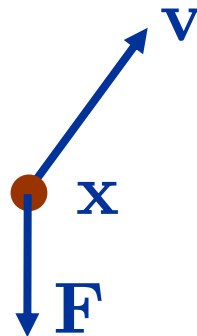


- introduction
- particle motion
- finite differences
- first order ODEs
- second order ODEs

# Particle Quantities



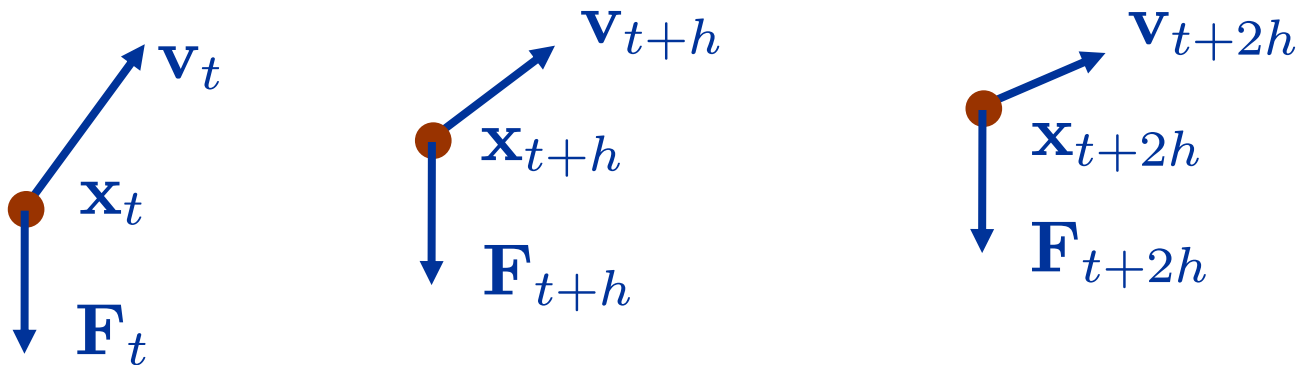
- quantities relevant for the motion of a particle:
  - mass  $m \in \mathbb{R}$
  - position  $\mathbf{x} \in \mathbb{R}^3$
  - velocity  $\mathbf{v} \in \mathbb{R}^3$
  - force  $\mathbf{F} \in \mathbb{R}^3$  acting on the particle
  - force  $\mathbf{F}$  can depend on position and velocity



# Particle Motion



- quantities are considered at discrete time points



- particle simulations are concerned with the computation of unknown future particle quantities  $\mathbf{x}_{t+h}$ ,  $\mathbf{v}_{t+h}$  using known current information  $\mathbf{x}_t$ ,  $\mathbf{v}_t$ ,  $\mathbf{F}_t$

# Governing Equation



- Newton's Second Law, Newton's motion equation, motion equation of a particle
- the force acting on an object is equal to the rate of change of its momentum (Impuls)

$$\mathbf{F}_t = \frac{d}{dt} (m\mathbf{v}_t) = \frac{dm}{dt} \mathbf{v}_t + m \frac{d\mathbf{v}_t}{dt}$$

- constant mass

$$\mathbf{F}_t = m \frac{d\mathbf{v}_t}{dt} = m \frac{d^2 \mathbf{x}_t}{dt^2}$$

# Governing Equation



- $\mathbf{F}_t = m \frac{d^2 \mathbf{x}_t}{dt^2}$  is an ordinary differential equation ODE
- describes the behavior of  $\mathbf{x}_t$  in terms of its derivatives with respect to time
- numerical integration can be employed to numerically solve the ODE, i. e. to approximate the unknown function  $\mathbf{x}_t$

# Governing Equation



- initial value problem of second order

$$\frac{d^2 \mathbf{x}_t}{dt^2} = \frac{1}{m} \mathbf{F}_t$$

- second-order ODEs can be rewritten as a system of two coupled equations of first order
- initial value problem of first order

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}_t$$

$$\frac{d\mathbf{v}_t}{dt} = \frac{1}{m} \mathbf{F}_t$$

# Initial Value Problem of First Order

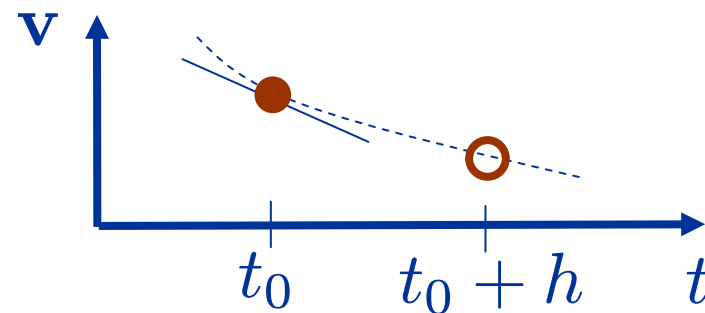
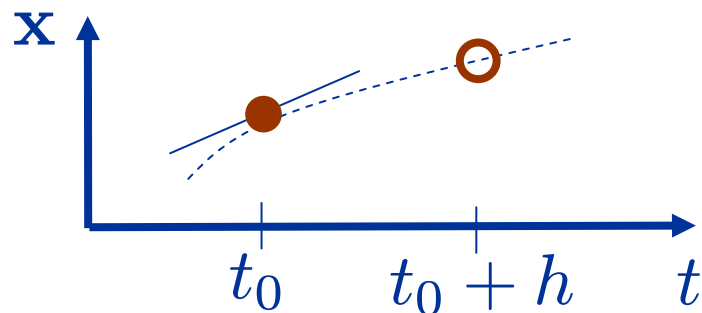


- functions  $\mathbf{x}_t$ ,  $\mathbf{v}_t$  represent the particle motion
- initial values are given  $\mathbf{x}_{t_0}$ ,  $\mathbf{v}_{t_0}$
- first-order differential equations are given

$$\frac{d\mathbf{v}_t}{dt} = \frac{1}{m} \mathbf{F}_t$$

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}_t$$

- at time  $t_0$ , the functions and their first derivatives are known
- how to compute  $\mathbf{x}_{t_0+h}$ ,  $\mathbf{v}_{t_0+h}$



# Outline



- introduction
- particle motion
- finite differences
- first order ODEs
- second order ODEs

# Finite Differences



- Taylor-series approximation

$$\mathbf{x}_{t+h} = \mathbf{x}_t + \frac{d\mathbf{x}_t}{dt} h + O(h^2)$$

$$\frac{d\mathbf{x}_t}{dt} = \frac{\mathbf{x}_{t+h} - \mathbf{x}_t}{h} + O(h)$$

- continuous ODEs are replaced with discrete finite-difference equations FDEs

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}_t \quad \Rightarrow \quad \frac{\mathbf{x}_{t+h} - \mathbf{x}_t}{h} = \mathbf{v}_t$$

$$\frac{d\mathbf{v}_t}{dt} = \frac{1}{m} \mathbf{F}_t \quad \Rightarrow \quad \frac{\mathbf{v}_{t+h} - \mathbf{v}_t}{h} = \frac{1}{m} \mathbf{F}_t$$

# Finite Differences



- polynomial fitting (line fitting in case of one sample)

$$\mathbf{x}_t = \mathbf{a}t + b$$

$$\Rightarrow \frac{d\mathbf{x}_t}{dt} = \mathbf{a} \quad \Rightarrow b = \mathbf{x}_t - \frac{d\mathbf{x}_t}{dt}t$$

$$\mathbf{x}_{t+h} = \frac{d\mathbf{x}_t}{dt}(t+h) + \mathbf{x}_t - \frac{d\mathbf{x}_t}{dt}t$$

which results in

$$\frac{d\mathbf{x}_t}{dt} = \frac{\mathbf{x}_{t+h} - \mathbf{x}_t}{h} + O(h)$$

# Finite Differences



- integral approach

$$dx_x = v_x dt$$

$$\int_{x_{x,t}}^{x_{x,t+h}} dx_x = x_{x,t+h} - x_{x,t} = \int_t^{t+h} v_x dt$$

$$x_{x,t+h} = x_{x,t} + \int_t^{t+h} v_x dt$$

$$x_{x,t+h} = x_{x,t} + h \frac{1}{2} (v_{x,t} + v_{x,t+h}) + O(h^2)$$

$$\frac{d\mathbf{x}_t}{dt} = \frac{\mathbf{x}_{t+h} - \mathbf{x}_t}{h} + O(h)$$

# Outline



- introduction
- particle motion
- finite differences
- first order ODEs
  - explicit approaches
  - predictor corrector approaches
  - implicit approaches
- second order ODEs

# Euler Method



$$\frac{d\mathbf{x}_t}{dt} = \dot{\mathbf{x}}_t = \mathbf{v}_t$$

$$\frac{d\mathbf{v}_t}{dt} = \dot{\mathbf{v}}_t = \frac{1}{m}\mathbf{F}_t$$

- initialize  $\mathbf{x}_{t_0} = \mathbf{x}_0$   $\mathbf{v}_{t_0} = \mathbf{v}_0$   $\mathbf{F}_{t_0}$   $m$   $h$
- numerical integration of position and velocity

$$\mathbf{x}_{t_0+h} = \mathbf{x}_{t_0} + h\dot{\mathbf{x}}_{t_0} = \mathbf{x}_{t_0} + h\mathbf{v}_{t_0}$$

$$\mathbf{v}_{t_0+h} = \mathbf{v}_{t_0} + h\dot{\mathbf{v}}_{t_0} = \mathbf{v}_{t_0} + h\frac{1}{m}\mathbf{F}_{t_0}$$

# Coupled Equations



- Euler step from  $t_0$  to  $t_0+h$

$$\mathbf{x}_{t_0+h} = \mathbf{x}_{t_0} + h\mathbf{v}_{t_0}$$

$$\mathbf{v}_{t_0+h} = \mathbf{v}_{t_0} + h\frac{1}{m}\mathbf{F}_{t_0}$$

- Euler step from  $t_0+h$  to  $t_0+2h$

$$\mathbf{x}_{t_0+2h} = \mathbf{x}_{t_0+h} + h\mathbf{v}_{t_0+h}$$

$$\mathbf{v}_{t_0+2h} = \mathbf{v}_{t_0+h} + h\frac{1}{m}\mathbf{F}_{t_0+h}$$

- the position update depends on velocity information
- the force update can depend on position and velocity

# Accuracy and Stability



- **discretization error** is defined as the difference between the solution of the ODE and the solution of the FDE
- the FDE is **consistent**, if the discretization error vanishes if the time step  $h$  approaches zero
- the FDE is **stable**, if previously introduced errors (discretization, round-off, mistakes) do not grow within a simulation step
- the FDE is **convergent**, if the solution of the FDE approaches the solution of the ODE

# *Accuracy and Stability*



- although the discretization error is diminished by smaller time steps in consistent schemes, the discretization error is introduced in each step of the FD scheme
- if previously introduced discretization errors are not amplified by the FD scheme, then it is stable
- consistent and stable schemes are convergent

# Stability



- if stability is influenced by the time step, the FD scheme is **conditionally stable**
- if the FD scheme is stable or unstable for arbitrary time steps, it is **unconditionally stable** or unstable
- ODE, FDE and the parameters influence the stability of a system

# *Time Step*



- larger time steps **do** speed up a simulation
- smaller time steps **can** improve the stability
- arbitrarily small time steps are not feasible due to round-off errors
  - for larger time steps, the error is dominated by the discretization error
  - for smaller time steps, the error is dominated by round-off errors
- performance of an FD scheme is trade-off between error order in terms of the time step and computing complexity

# Improving the Accuracy Leap Frog



$$\mathbf{v}_{t+\frac{h}{2}} = \mathbf{v}_{t-\frac{h}{2}} + h \frac{1}{m} \mathbf{F}_t$$

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h \mathbf{v}_{t+\frac{h}{2}}$$

- error order  $O(h^2)$
- can generally handle larger time steps  $h$  compared to Euler

Euler

```
...  
computeForces(); //F(t)  
positionEuler(h); //x=x(t+h)=x(t)+hv(t)  
velocityEuler(h); //v=v(t+h)=v(t)+ha(t)  
...
```

Leap Frog

```
initV() // v(0) = v(0) - (h/2)a(0)  
...  
computeForces(); //F(t)  
velocityEuler(h); //v=v(t+h)=v(t)+ha(t)  
positionEuler(h); //x=x(t+h)=x(t)+hv(t+h)  
...
```

# Euler-Cromer



$$\mathbf{v}_{t+h} = \mathbf{v}_t + h \frac{1}{m} \mathbf{F}_t$$

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h \mathbf{v}_{t+h}$$

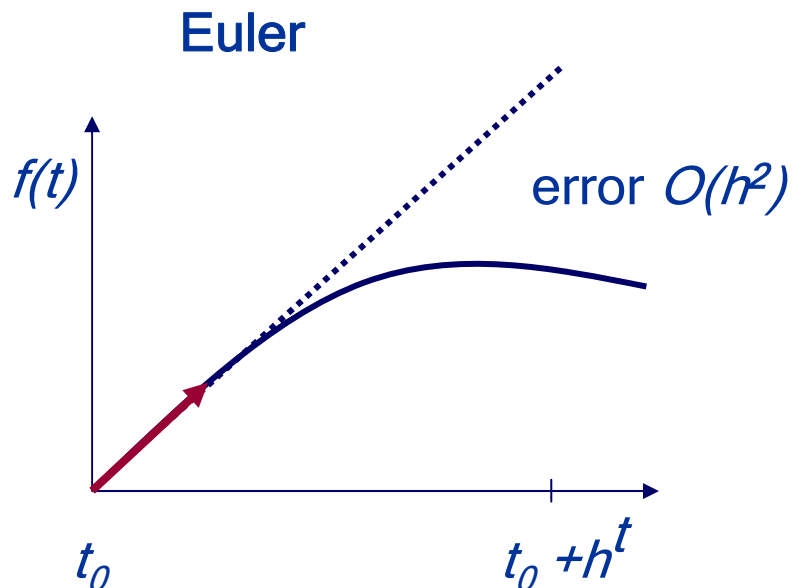
## Euler

```
...  
computeForces(); //F(t)  
positionEuler(h); //x=x(t+h)=x(t)+hv(t)  
velocityEuler(h); //v=v(t+h)=v(t)+ha(t)  
...
```

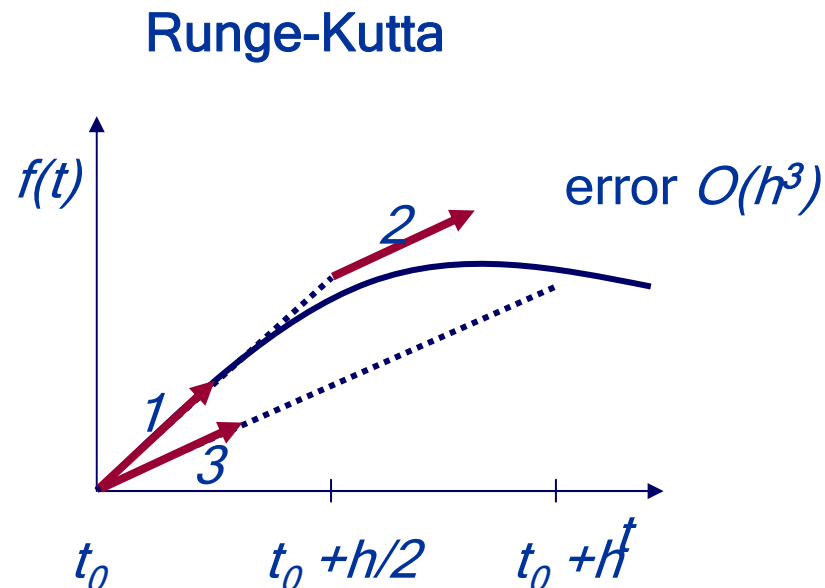
## Euler-Cromer

```
...  
computeForces(); //F(t)  
velocityEuler(h); //v=v(t+h)=v(t)+ha(t)  
positionEuler(h); //x=x(t+h)=x(t)+hv(t+h)  
...
```

# Runge-Kutta



- compute the derivative at  $t_0$
- approximate  $f(t_0 + h)$  using the derivative at  $t_0$



- compute the derivative at  $t_0$
- compute  $f(t_0 + h/2)$
- compute the derivative at  $t_0 + h/2$
- approximate  $f(t_0 + h)$  using the derivative at  $t_0 + h/2$

# Second-Order Runge-Kutta Midpoint Method



$$\dot{\mathbf{x}}_t = \mathbf{v}_t$$

$$\dot{\mathbf{v}}_{\mathbf{x}_t, \mathbf{v}_t} = \frac{1}{m} \mathbf{F}_t$$

- compute  $\mathbf{x}'(t)$
- compute  $\mathbf{v}'(t)$
- compute  $\mathbf{x}'(t+h/2)$
- compute  $\mathbf{v}'(t+h/2)$   
with  $\mathbf{x}(t+h/2)$  and  $\mathbf{v}(t+h/2)$
- compute  $\mathbf{x}(t+h)$  with  $\mathbf{x}'(t+h/2)$
- compute  $\mathbf{v}(t+h)$  with  $\mathbf{v}'(t+h/2)$

$$\mathbf{k}_1 = \dot{\mathbf{x}}_t$$

$$\mathbf{l}_1 = \dot{\mathbf{v}}_{\mathbf{x}_t, \mathbf{v}_t}$$

$$\mathbf{k}_2 = \dot{\mathbf{x}}_t + \mathbf{l}_1 \frac{h}{2}$$

$$\mathbf{l}_2 = \dot{\mathbf{v}}_{\mathbf{x}_t + \mathbf{k}_1 \frac{h}{2}, \mathbf{k}_2}$$

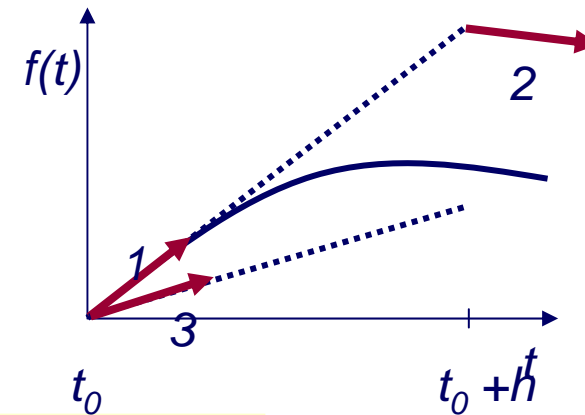
$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{k}_2$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + h\mathbf{l}_2$$

# Second-Order Runge-Kutta Heun

$$\dot{\mathbf{x}}_t = \mathbf{v}_t$$

$$\dot{\mathbf{v}}_{\mathbf{x}_t, \mathbf{v}_t} = \frac{1}{m} \mathbf{F}_t$$



- compute  $\mathbf{x}'(t)$
- compute  $\mathbf{v}'(t)$
- compute  $\mathbf{x}'(t+h)$
- compute  $\mathbf{v}'(t+h)$
- c.  $\mathbf{x}(t+h)$  with  $\mathbf{x}'(t)$  and  $\mathbf{x}'(t+h)$
- c.  $\mathbf{v}(t+h)$  with  $\mathbf{v}'(t)$  and  $\mathbf{v}'(t+h)$

$$\mathbf{k}_1 = \dot{\mathbf{x}}_t$$

$$\mathbf{l}_1 = \dot{\mathbf{v}}_{\mathbf{x}_t, \mathbf{v}_t}$$

$$\mathbf{k}_2 = \dot{\mathbf{x}}_t + \mathbf{l}_1 h$$

$$\mathbf{l}_2 = \dot{\mathbf{v}}_{\mathbf{x}_t + \mathbf{k}_1 h, \mathbf{k}_2}$$

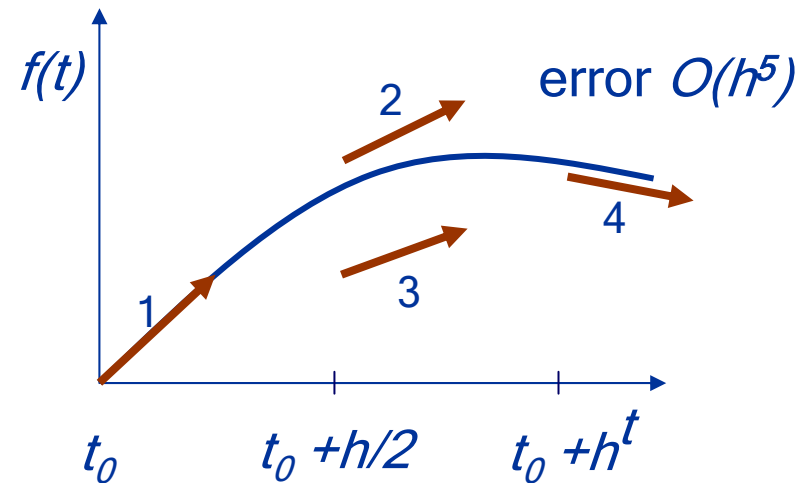
$$\mathbf{x}_{t+h} = \mathbf{x}_t + h \frac{\mathbf{k}_1 + \mathbf{k}_2}{2}$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + h \frac{\mathbf{l}_1 + \mathbf{l}_2}{2}$$

# Fourth-Order Runge-Kutta



- compute  $f'(t_0)$  (1)
- compute  $f(t_0+h/2)$  with  $f(t_0)$  and  $f'(t_0)$
- compute  $f'(t_0+h/2)$  (2)
- compute  $f(t_0+h)$  with  $f(t_0)$  and  $f'(t_0+h/2)$
- compute  $f'(t_0+h)$  (3)
- compute  $f(t_0+h)$  with  $f(t_0)$  and a weighted average of all derivatives (1) - (4)



# Fourth-Order Runge-Kutta



- four derivative computations per time step
- error  $O(h^5)$

$$\mathbf{k}_1 = \dot{\mathbf{x}}_t$$

$$\mathbf{l}_1 = \dot{\mathbf{v}}_{\mathbf{x}_t, \mathbf{v}_t}$$

$$\mathbf{k}_2 = \dot{\mathbf{x}}_t + \mathbf{l}_1 \frac{h}{2}$$

$$\mathbf{l}_2 = \dot{\mathbf{v}}_{\mathbf{x}_t + \mathbf{k}_1 \frac{h}{2}, \mathbf{k}_2}$$

$$\mathbf{k}_3 = \dot{\mathbf{x}}_t + \mathbf{l}_2 \frac{h}{2}$$

$$\mathbf{l}_3 = \dot{\mathbf{v}}_{\mathbf{x}_t + \mathbf{k}_2 \frac{h}{2}, \mathbf{k}_3}$$

$$\mathbf{k}_4 = \dot{\mathbf{x}}_t + \mathbf{l}_3 h$$

$$\mathbf{l}_4 = \dot{\mathbf{v}}_{\mathbf{x}_t + \mathbf{k}_3 h, \mathbf{k}_4}$$

$$\mathbf{x}_{t+h} = \mathbf{x}_t + \frac{h}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + \frac{h}{6} (\mathbf{l}_1 + 2\mathbf{l}_2 + 2\mathbf{l}_3 + \mathbf{l}_4)$$

# Performance



## Euler

- one computation of the derivative per time step
- error  $O(h^2)$

## Runge-Kutta

- two (four) computations of the derivative per time step
- error  $O(h^3)$ ,  $O(h^5)$
- allows larger time steps

- similar performance if the time step for RK 2 is twice the time step for Euler
- does RK allow for faster simulations than Euler?

$$0 \leq \text{error} < \frac{h^n}{n!} f_t^{(n)}$$

# *Implementation*



- Euler
  - one function evaluation
  - force computation
  - position and velocity update
- Runge-Kutta
  - multiple function evaluations
  - computation of auxiliary forces, positions, velocities
    - once for second order
    - three times for fourth order
  - position and velocity update

# Outline



- introduction
- particle motion
- finite differences
- first order ODEs
  - explicit approaches
  - predictor corrector approaches
  - implicit approaches
- second order ODEs

# Predictor-Corrector Methods



- predict a value from current (and previous) derivatives
- correct the predicted value with its derivative
- second-order Adams-Bashforth predictor

$$f_{t+h} = f_t + \frac{h}{2} \left( 3\dot{f}_t - \dot{f}_{t-h} \right) + O(h^3)$$

- second-order Adams-Moulton corrector

$$f_{t+h} = f_t + \frac{h}{2} \left( \dot{f}_{t+h} + \dot{f}_t \right) + O(h^3)$$

- based on Lagrange polynomials
- can be efficiently implemented with two derivative computations per simulation step
- requires values at previous time steps, not self-starting

# Adams-Bashforth Predictors



$$f_{t+h} = f_t + hf_t + O(h^2)$$

$$f_{t+h} = f_t + \frac{h}{2} (3\dot{f}_t - \dot{f}_{t-h}) + O(h^3)$$

$$f_{t+h} = f_t + \frac{h}{12} (23\dot{f}_t - 16\dot{f}_{t-h} + 5\dot{f}_{t-2h}) + O(h^4)$$

$$f_{t+h} = f_t + \frac{h}{24} (55\dot{f}_t - 59\dot{f}_{t-h} + 37\dot{f}_{t-2h} - 9\dot{f}_{t-3h}) + O(h^5)$$

...

$$f_{t+h} = f_t + \frac{h}{720} (1901\dot{f}_t - 2774\dot{f}_{t-h} + 2616\dot{f}_{t-2h} - 1274\dot{f}_{t-3h} + 251\dot{f}_{t-4h})$$

...

# Adams-Moulton Correctors



$$f_{t+h} = f_t + h\dot{f}_{t+h} + O(h^2)$$

$$f_{t+h} = f_t + \frac{h}{2} \left( \dot{f}_{t+h} + \dot{f}_t \right) + O(h^3)$$

$$f_{t+h} = f_t + \frac{h}{12} \left( 5\dot{f}_{t+h} + 8\dot{f}_t - \dot{f}_{t-h} \right) + O(h^4)$$

$$f_{t+h} = f_t + \frac{h}{24} \left( 9\dot{f}_{t+h} + 19\dot{f}_t - 5\dot{f}_{t-h} + \dot{f}_{t-2h} \right) + O(h^5)$$

...

$$f_{t+h} = f_t + \frac{h}{720} \left( 251\dot{f}_{t+h} + 646\dot{f}_t - 264\dot{f}_{t-h} + 106\dot{f}_{t-2h} - 19\dot{f}_{t-3h} \right)$$

...

# Richardson Extrapolation



- $f(h)$  approximates  $a_0$ , error is a series in powers of  $h$

$$f(h) = a_0 + a_1 h^{p_1} + a_2 h^{p_2} + \dots$$

$$f\left(\frac{h}{2}\right) = a_0 + a_1 \frac{h^{p_1}}{2^{p_1}} + a_2 \frac{h^{p_2}}{2^{p_2}} + \dots$$

- eliminating  $a_1$  results in

$$a_0 = \frac{2^{p_1} f\left(\frac{h}{2}\right) - f(h)}{2^{p_1} - 1} + a'_2 h^{p_2} + \dots$$

- error is reduced from order  $h^{p_1}$  to  $h^{p_2}$
- can generally be applied to integration schemes to improve the accuracy

# Outline



- introduction
- particle motion
- finite differences
- first order ODEs
  - explicit approaches
  - predictor corrector approaches
  - implicit approaches
- second order ODEs

# Explicit and Implicit Integration



## explicit Euler

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_t$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + h\frac{1}{m}\mathbf{F}_t$$

- one unknown per equation
- direct calculation of  $\mathbf{x}(t+h)$  and  $\mathbf{v}(t+h)$
- non-linear equations have no effect on the approach
- can handle non-analytical, procedural forces

## implicit Euler

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_{t+h}$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + h\frac{1}{m}\mathbf{F}_{t+h}$$

- system of algebraic equations with many unknowns
- simultaneous computation of  $\mathbf{x}(t+h)$  and  $\mathbf{v}(t+h)$
- solution of a system of equations
- non-linear equations are commonly linearized to get a system of linear equations

# Linearization of Scalar Functions



- one-dimensional scalar field

$$f_{x+h} = f_x + hf'_x + O(h^2)$$

- multi-dimensional scalar field

$$f_{\mathbf{x}+\mathbf{h}} = f_{\mathbf{x}} + \mathbf{h} \cdot \text{grad } f_{\mathbf{x}} + O(\|\mathbf{h}\|^2)$$

- gradient

$$\text{grad } f_{\mathbf{x}} = \nabla f_{\mathbf{x}} = \left( \frac{\partial f_{\mathbf{x}}}{\partial x_1}, \frac{\partial f_{\mathbf{x}}}{\partial x_2}, \dots, \frac{\partial f_{\mathbf{x}}}{\partial x_n} \right)^T$$

- nabla, del

$$\nabla = \left( \frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \dots, \frac{\partial}{\partial x_n} \right)^T$$

# Linearization of Vector Fields



- multi-dimensional vector field

$$\mathbf{F}_{\mathbf{x}+\mathbf{h}} = \mathbf{F}_{\mathbf{x}} + \mathbf{J}_{F_{\mathbf{x}}}\mathbf{h} + O(\|\mathbf{h}\|^2)$$

- Jacobi matrix

$$\mathbf{J}_{F_{\mathbf{x}}} = \frac{\partial \mathbf{F}_{\mathbf{x}}}{\partial \mathbf{x}} = \begin{pmatrix} \text{grad}^T F_{1,\mathbf{x}} \\ \dots \\ \text{grad}^T F_{m,\mathbf{x}} \end{pmatrix} = \begin{pmatrix} \frac{\partial F_{1,\mathbf{x}}}{\partial x_1} & \dots & \frac{\partial F_{1,\mathbf{x}}}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_{m,\mathbf{x}}}{\partial x_1} & \dots & \frac{\partial F_{m,\mathbf{x}}}{\partial x_n} \end{pmatrix}$$

# *Implicit Integration Theta Scheme*



$$\mathbf{x}_{t+h} = \mathbf{x}_t + h \left( (1 - \theta) \cdot \mathbf{v}_t + \theta \cdot \mathbf{v}_{t+h} \right)$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + h \left( (1 - \theta) \cdot \frac{\mathbf{F}_t}{m} + \theta \cdot \frac{\mathbf{F}_{t+h}}{m} \right)$$

- explicit Euler  $\theta = 0$
- implicit Euler  $\theta = 1$
- Crank Nicolson  $\theta = 0.5$

# Theta Scheme Implementation



- rewriting the problem for  $\theta = 0.5$

$$m\mathbf{v}_{t+h} = m\mathbf{v}_t + \frac{h}{2} \left( \mathbf{F}_{\mathbf{x}_t} + \mathbf{F}_{\mathbf{x}_t + \frac{h}{2}(\mathbf{v}_t + \mathbf{v}_{t+h})} \right)$$

- force linearization

$$\mathbf{F}_{\mathbf{x}_t + \frac{h}{2}(\mathbf{v}_t + \mathbf{v}_{t+h})} \approx \mathbf{F}_{\mathbf{x}_t} + \frac{\partial \mathbf{F}_{\mathbf{x}_t}}{\partial \mathbf{x}} \cdot \frac{h}{2} \cdot (\mathbf{v}_t + \mathbf{v}_{t+h})$$

- solving a linear system for  $\mathbf{v}(t+h)$

$$\left[ m - \frac{h^2}{4} \cdot \frac{\partial \mathbf{F}_{\mathbf{x}_t}}{\partial \mathbf{x}} \right] \cdot \mathbf{v}_{t+h} \approx m \cdot \mathbf{v}_t + h \cdot \mathbf{F}_{\mathbf{x}_t} + \frac{\partial \mathbf{F}_{\mathbf{x}_t}}{\partial \mathbf{x}} \cdot \frac{h^2}{4} \mathbf{v}_t$$

# Theta Scheme

## Conjugate Gradient



- linear system

$$\mathbf{A} \cdot \mathbf{v} = \mathbf{b}$$

- gradient of a function

$$\nabla \mathbf{f}(\mathbf{v}) = \mathbf{A} \cdot \mathbf{v} - \mathbf{b}$$

with

$$\nabla \mathbf{f}(\mathbf{v}) = 0$$

- iterative solution for  $\mathbf{v}$  with initial value  $\mathbf{v}_0$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha \cdot \mathbf{w}(\nabla \mathbf{f}(\mathbf{v}_k))$$

# Conjugate Gradient Method



- $\mathbf{v}_0 = \mathbf{v}(t)$
- direction  $\mathbf{d}$
- residual  $\mathbf{r}$
- step size  $\alpha$
- $\mathbf{v}(t+h) = \mathbf{v}_j$
  
- $\mathbf{A}$  is symmetric, positive-definite
- $-\mathbf{r}_0, -\mathbf{d}_0$  gradient of function  $f$
- $\mathbf{d}_i, \mathbf{d}_j$  are conjugate,  
i.e.  $\mathbf{d}_i^T \mathbf{A} \mathbf{d}_j = 0$
- $\mathbf{r}_i = 0$  in maximal  $n$  steps,  
if  $\mathbf{v}$  has  $n$  components

$$\mathbf{d}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{v}_0$$

$$\alpha = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_i^T \mathbf{A} \mathbf{d}_i}$$

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \alpha \mathbf{d}_i$$

$$\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha \mathbf{A} \mathbf{d}_i$$

$$\mathbf{d}_{i+1} = \mathbf{r}_{i+1} + \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i} \mathbf{d}_i$$

# Outline



- introduction
- particle motion
- finite differences
- first order ODEs
- second order ODEs

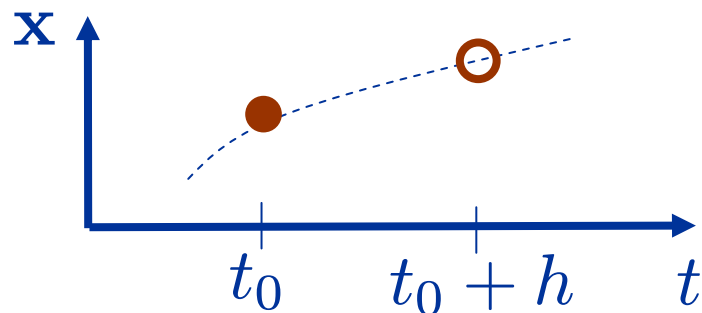
# Initial Value Problem of Second Order



- function  $\mathbf{x}_t$  represents the particle motion
- initial values are given  $\mathbf{x}_{t_0}, (\dot{\mathbf{x}}_{t_0})$
- second-order differential equation is given

$$\frac{d^2 \mathbf{x}_t}{dt^2} = \frac{1}{m} \mathbf{F}_t$$

- at time  $t_0$ , the function and their derivatives are known
- how to compute  $\mathbf{x}_{t_0+h}, (\dot{\mathbf{x}}_{t_0+h})$



# Overview of Integration Schemes



integration methods for  
first-order ODEs

integration methods for  
second-order ODE  
(Newton's motion equation)

Euler, Euler-Cromer  
Heun, Midpoint-Method  
Runge-Kutta, Leap-Frog

Verlet, velocity Verlet  
Beeman  
Gear

# Verlet



- Taylor approximations of  $\mathbf{x}_{t+h}$  and  $\mathbf{x}_{t-h}$

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_t + \frac{h^2}{2} \frac{\mathbf{F}_t}{m} + \frac{h^3}{6} \mathbf{x}_t^{(3)} + O(h^4)$$

$$\mathbf{x}_{t-h} = \mathbf{x}_t - h\mathbf{v}_t + \frac{h^2}{2} \frac{\mathbf{F}_t}{m} - \frac{h^3}{6} \mathbf{x}_t^{(3)} + O(h^4)$$

- adding both approximations

$$\mathbf{x}_{t+h} = 2\mathbf{x}_t - \mathbf{x}_{t-h} + h^2 \frac{\mathbf{F}_t}{m} + O(h^4)$$

# Verlet



- independent of velocity
- one derivative computation per time step
- efficient to compute, comparatively accurate
- third-order in the position
- if required, velocity can be computed a posteriori with a central difference

$$\mathbf{v}_t = \frac{\mathbf{x}_{t+h} - \mathbf{x}_{t-h}}{2h} + O(h^2)$$

- velocity is commonly required for collision response or damping

# Velocity Verlet



- one force (derivative) computation per time step
- second-order accuracy in position and velocity

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_t + \frac{h^2}{2} \frac{\mathbf{F}_t}{m} + O(h^3)$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + \frac{h}{2} \left( \frac{\mathbf{F}_t}{m} + \frac{\mathbf{F}_{t+h}}{m} \right) + O(h^3)$$

$\mathbf{F}_{t+h}$  is computed  
using  $\mathbf{x}_{t+h}$

- equivalent to

$$\mathbf{v}_{t+\frac{h}{2}} = \mathbf{v}_t + \frac{h}{2} \frac{\mathbf{F}_t}{m}$$

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_{t+\frac{h}{2}}$$

$$\mathbf{v}_{t+h} = \mathbf{v}_{t+\frac{h}{2}} + \frac{h}{2} \frac{\mathbf{F}_{t+h}}{m}$$

# Beeman



- one force (derivative) computation per time step
- efficient to compute
- third-order accuracy in position and velocity

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_t + h^2 \left( \frac{2}{3} \frac{\mathbf{F}_t}{m} - \frac{1}{6} \frac{\mathbf{F}_{t-h}}{m} \right) + O(h^4)$$

$$\mathbf{v}_{t+h} = \mathbf{v}_t + h \left( \frac{5}{12} \frac{\mathbf{F}_{t+h}}{m} + \frac{2}{3} \frac{\mathbf{F}_t}{m} - \frac{1}{12} \frac{\mathbf{F}_{t-h}}{m} \right) + O(h^4)$$

- $\mathbf{F}_{t+h}$  is computed using  $\mathbf{x}_{t+h}$

# Gear Integration



position    velocity    acceleration    jerk



$$\mathbf{x}_{t+h} = \mathbf{x}_t + \frac{\dot{\mathbf{x}}_t}{1!} h + \frac{\ddot{\mathbf{x}}_t}{2!} h^2 + \frac{\mathbf{x}_t^{(3)}}{3!} h^3 + \frac{\mathbf{x}_t^{(4)}}{4!} h^4 + \frac{\mathbf{x}_t^{(5)}}{5!} h^5 + \dots$$

$$\mathbf{r}_{t+h}^0 = \mathbf{r}_t^0 + \mathbf{r}_t^1 + \mathbf{r}_t^2 + \mathbf{r}_t^3 + \mathbf{r}_t^4 + \mathbf{r}_t^5 + \dots$$

$$\mathbf{r}_t^k = \left( \frac{d^k \mathbf{x}_t}{dt^k} \right) \left( \frac{h^k}{k!} \right)$$

# Gear Integration



$$\mathbf{x}_{t+h} = \mathbf{x}_t + \frac{\dot{\mathbf{x}}_t}{1!}h + \frac{\ddot{\mathbf{x}}_t}{2!}h^2 + \frac{\mathbf{x}_t^{(3)}}{3!}h^3 + \frac{\mathbf{x}_t^{(4)}}{4!}h^4 + \frac{\mathbf{x}_t^{(5)}}{5!}h^5 + \dots$$

$$\mathbf{r}_{t+h}^0 = \mathbf{r}_t^0 + \mathbf{r}_t^1 + \mathbf{r}_t^2 + \mathbf{r}_t^3 + \mathbf{r}_t^4 + \mathbf{r}_t^5 + \dots$$

$$h\dot{\mathbf{x}}_{t+h} = h\dot{\mathbf{x}}_t + h\frac{\ddot{\mathbf{x}}_t}{1!}h + h\frac{\mathbf{x}_t^{(3)}}{2!}h^2 + h\frac{\mathbf{x}_t^{(4)}}{3!}h^3 + h\frac{\mathbf{x}_t^{(5)}}{4!}h^4 + \dots$$

$$\mathbf{r}_{t+h}^1 = \mathbf{r}_t^1 + 2\mathbf{r}_t^2 + 3\mathbf{r}_t^3 + 4\mathbf{r}_t^4 + 5\mathbf{r}_t^5 + \dots$$

$$\frac{h^2}{2}\ddot{\mathbf{x}}_{t+h} = \frac{h^2}{2}\ddot{\mathbf{x}}_t + \frac{h^2}{2}\frac{\mathbf{x}_t^{(3)}}{1!}h + \frac{h^2}{2}\frac{\mathbf{x}_t^{(4)}}{2!}h^2 + \frac{h^2}{2}\frac{\mathbf{x}_t^{(5)}}{3!}h^3 + \dots$$

$$\mathbf{r}_{t+h}^2 = \mathbf{r}_t^2 + 3\mathbf{r}_t^3 + 6\mathbf{r}_t^4 + 10\mathbf{r}_t^5 + \dots$$

# Gear - Prediction



$$\mathbf{x}_{t+h} = \mathbf{r}_{t+h}^0 = \mathbf{r}_t^0 + \mathbf{r}_t^1 + \mathbf{r}_t^2 + \mathbf{r}_t^3 + \mathbf{r}_t^4 + \mathbf{r}_t^5$$

$$h\mathbf{v}_{t+h} = \mathbf{r}_{t+h}^1 = \mathbf{r}_t^1 + 2\mathbf{r}_t^2 + 3\mathbf{r}_t^3 + 4\mathbf{r}_t^4 + 5\mathbf{r}_t^5$$

$$\frac{h^2}{2}\mathbf{a}_{t+h} = \mathbf{r}_{t+h}^2 = \mathbf{r}_t^2 + 3\mathbf{r}_t^3 + 6\mathbf{r}_t^4 + 10\mathbf{r}_t^5$$

$$\mathbf{r}_{t+h}^3 = \mathbf{r}_t^3 + 4\mathbf{r}_t^4 + 10\mathbf{r}_t^5$$

$$\mathbf{r}_{t+h}^4 = \mathbf{r}_t^4 + 5\mathbf{r}_t^5$$

$$\mathbf{r}_{t+h}^5 = \mathbf{r}_t^5$$

# Gear - Correction



- error

$$\mathbf{error}_{t+h} = \mathbf{r}_{t+h}^2 - \frac{\mathbf{F}_{t+h}}{m} \frac{h^2}{2!}$$

$$\mathbf{r}_{t+h}^k = \mathbf{r}_{t+h}^k - C_k \mathbf{error}_{t+h}$$

- error correction coefficients  $C_k$

k = 0	k = 1	k = 2	k = 3	k = 4	k = 5
$\frac{3}{20}$	$\frac{251}{360}$	1	$\frac{11}{18}$	$\frac{1}{6}$	$\frac{1}{60}$

# Gear - Implementation



- initialization

$$\mathbf{r}_0^0 = \mathbf{x}_0$$

$$\mathbf{r}_0^2 = \frac{\mathbf{F}_0}{m} \frac{h^2}{2}$$

$$\mathbf{r}_0^1 = \mathbf{v}_0 h$$

$$\mathbf{r}_0^3 = \mathbf{r}_0^4 = \mathbf{r}_0^5 = 0$$

- integration

- prediction
- error estimation
- correction

$$\mathbf{r}_{t+h}^0 = \mathbf{r}_t^0 + \dots + \mathbf{r}_t^5$$

$$\mathbf{r}_{t+h}^i = \dots$$

$$\mathbf{error}_{t+h} = \mathbf{r}_{t+h}^2 - \frac{\mathbf{F}_{t+h}}{m} \frac{h^2}{2!}$$

$$\mathbf{r}_{t+h}^k = \mathbf{r}_{t+h}^k - C_k \mathbf{error}_{t+h}$$

# Comparison – Explicit Schemes



method	force comp. per time step	error position	error velocity
Euler	1	1	1
RK 2	2	2	2
RK 4	4	4	4
Leap-Frog	1	2	2
Verlet	1	3	1
Velocity Verlet	1	2	2
Beeman	1	3	3

# Comparison – *Explicit Schemes*



- “first-order” methods
  - accuracy corresponds to computing complexity
  - position and velocity have the same error order
- “second-order” methods
  - improved accuracy with minimal computing complexity
  - error order might differ for position and velocity
- implicit methods cannot be compared this way
  - do not only compute forces (derivatives)
  - commonly require to solve a linear system
  - improved stability even for low error orders, impl. Euler with error order one can be unconditionally stable, e. g. for harmonic oscillators (springs)

# *Advantages / Disadvantages*



- explicit methods
  - simple to set up and program
  - fast computation per integration step
  - suitable for parallel architectures
  - small time steps required for stability
  - many computing steps required for a given time interval  $t$
- implicit methods
  - stability is maintained for large time steps
  - require less steps for a given interval  $t$
  - large time steps can cause large truncation errors
  - complicate to set up
  - less flexible, problems with non-analytical forces
  - large computing time per integration step

# *Advantages / Disadvantages*



- predictor-corrector methods
  - not self-starting
  - have to be re-initialized in case of discontinuities, e. g. due to collision response
- in general, implicit methods are more robust (stable) compared to explicit methods
  - if an explicit scheme is not conditionally or unconditionally stable it cannot be used in applications regardless of its efficiency
- explicit methods can be computed efficiently which is essential if frequent updates are required
  - if an implicit scheme cannot be computed at interactive rates, it cannot be used in interactive applications regardless of the time step

# Summary



- motion equation for a mass point
  - second-order differential equation
  - coupled system of first-order differential equations
- numerical integration
  - initial values  $\mathbf{x}_t$  and  $\mathbf{v}_t$
  - approximate integration of  $\mathbf{v}$  and  $\mathbf{x}$  through time with time step  $h$
- integration schemes
  - Euler, Euler-Cromer, Leap-Frog, Heun, Runge-Kutta
  - Crank-Nicolson, implicit Euler
  - Gear, Verlet, velocity Verlet, Beeman
- trade-off between accuracy and computing cost
- goal: maximizing the ratio of time step and computing cost