

Dr Fangcheng Zhong

Physics Simulation

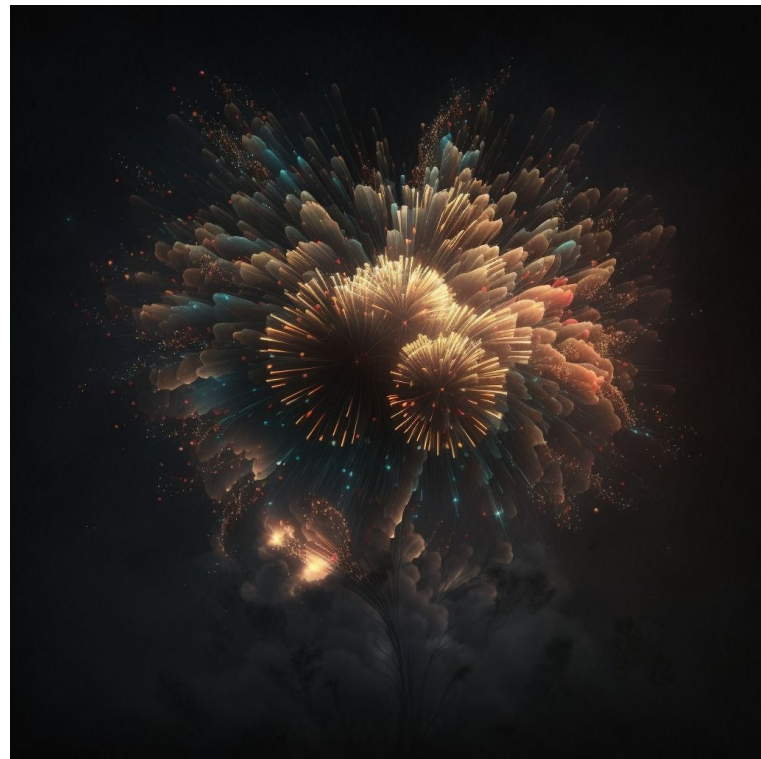


UNIVERSITY OF
CAMBRIDGE

Outline

- Physically-based simulation
 - Particle system
 - Rigid-body dynamics
 - Mass-spring system
 - Fluid dynamics
- Numerical solvers for differential equations
 - Finite difference methods
 - Euler method, Runge-Kutta method, trapezoidal rule
 - truncation error, convergence, stability

Particle System



Particle System

initialisation

$$\mathbf{x}(0), \mathbf{x}'(0)$$

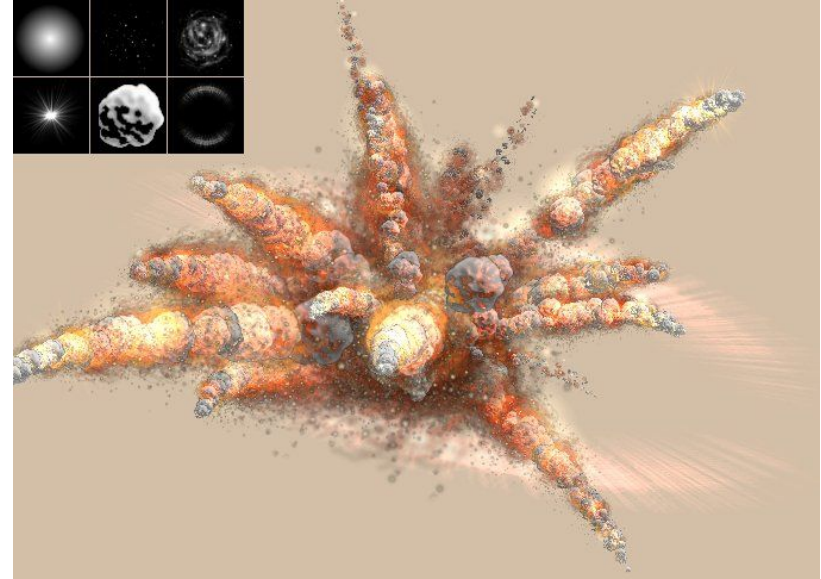
simulation

$$\mathbf{f} = m\mathbf{a} = m\mathbf{x}''$$

Shape		Drag Coefficient
Sphere	→ 	0.47
Half-sphere	→ 	0.42
Cone	→ 	0.50
Cube	→ 	1.05
Angled Cube	→ 	0.80
Long Cylinder	→ 	0.82
Short Cylinder	→ 	1.15
Streamlined Body	→ 	0.04
Streamlined Half-body	→ 	0.09

Measured Drag Coefficients

rendering



<https://codepen.io/rainner/pen/LREdXd>

Rigid Body Dynamics



Rigid Body Dynamics

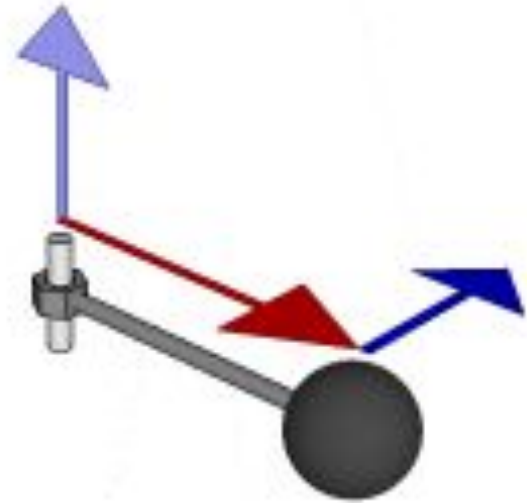
linear acceleration

$$\mathbf{f} = m\mathbf{a} = m\mathbf{x}''$$

angular acceleration

$$\boldsymbol{\tau} = mr^2\boldsymbol{\alpha} = mr^2\boldsymbol{\omega}'$$

$$\begin{aligned}\boldsymbol{\tau} &= \mathbf{r} \times \mathbf{F} \\ \mathbf{L} &= \mathbf{r} \times \mathbf{p}\end{aligned}$$



Deformation

elastic



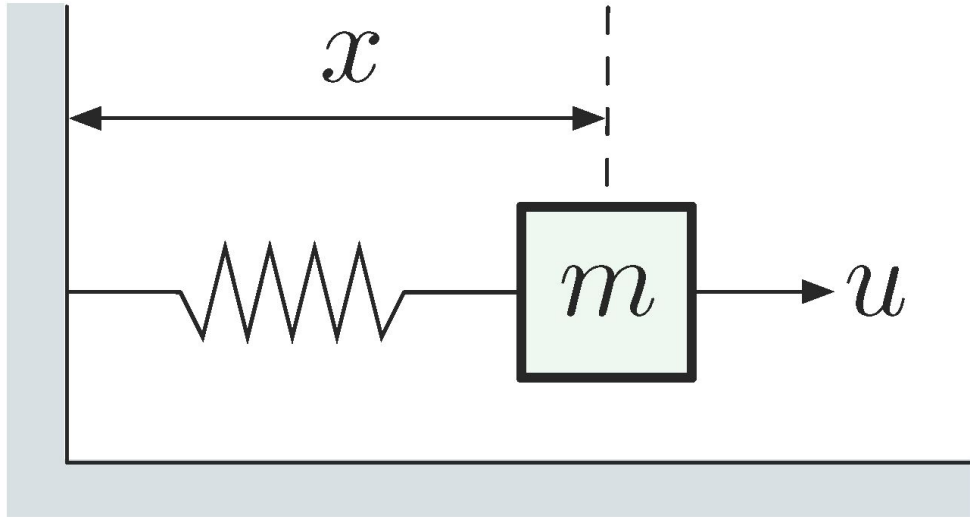
plastic



both



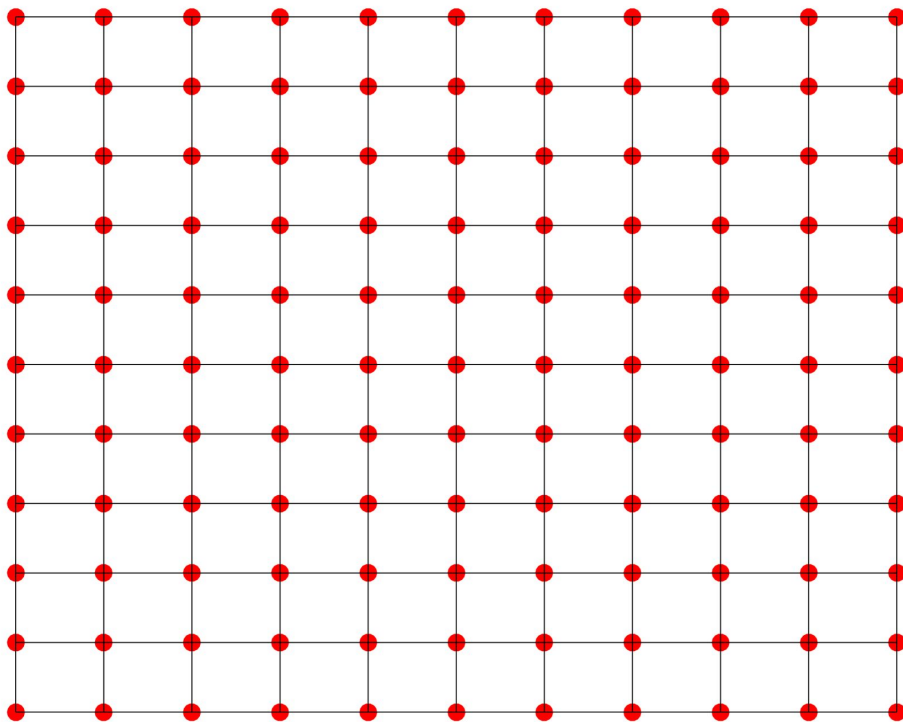
Mass-Spring System



Hooke's law

$$f = -k (\|x\| - l) = mx''$$

Mass-Spring System



cloth simulation

WebGL demo ([link1](#)) ([link2](#))



Fluid Dynamics



Lagrangian
vs
Eulerian
specification



Finite Difference Methods

- Approximate derivatives with finite differences to solve differential equations

$$F(t, x(t), x'(t), \dots, x^{(n)}(t)) = 0$$

general-form ordinary differential equation (ODE)

Truncation Error

Taylor expansion

$$f(x) = f(a) + f'(a)h + \frac{f''(a)}{2!}h^2 + \frac{f'''(a)}{3!}h^3 + \dots \frac{f^{(n)}(a)}{n!}h^n + \dots$$

$$h = (x - a)$$

$$\underbrace{\frac{f(a+h) - f(a)}{h}}_{\text{1st order approximation}} = f'(a) + \underbrace{\frac{f''(a)}{2!}h + \frac{f'''(a)}{3!}h^2 + \dots \frac{f^{(n)}(a)}{n!}h^{n-1} + \dots}_{\text{truncation error}}$$

Finite Difference Methods

finite difference coefficients

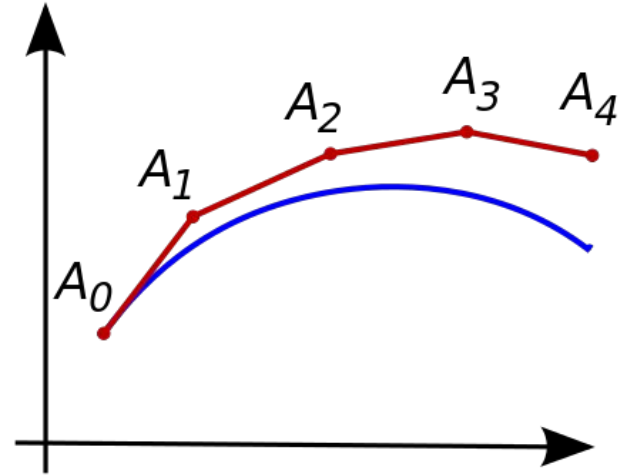
Derivative	Accuracy	-4	-3	-2	-1	0	1	2	3	4
1	2				$-1/2$	0	$1/2$			
	4			$1/12$	$-2/3$	0	$2/3$	$-1/12$		
	6		$-1/60$	$3/20$	$-3/4$	0	$3/4$	$-3/20$	$1/60$	
	8	$1/280$	$-4/105$	$1/5$	$-4/5$	0	$4/5$	$-1/5$	$4/105$	$-1/280$
2	2				1	-2	1			
	4			$-1/12$	$4/3$	$-5/2$	$4/3$	$-1/12$		
	6		$1/90$	$-3/20$	$3/2$	$-49/18$	$3/2$	$-3/20$	$1/90$	
	8	$-1/560$	$8/315$	$-1/5$	$8/5$	$-205/72$	$8/5$	$-1/5$	$8/315$	$-1/560$
3	2			$-1/2$	1	0	-1	$1/2$		
	4		$1/8$	-1	$13/8$	0	$-13/8$	1	$-1/8$	
	6	$-7/240$	$3/10$	$-169/120$	$61/30$	0	$-61/30$	$169/120$	$-3/10$	$7/240$
4	2			1	-4	6	-4	1		
	4		$-1/6$	2	$-13/2$	$28/3$	$-13/2$	2	$-1/6$	
	6	$7/240$	$-2/5$	$169/60$	$-122/15$	$91/8$	$-122/15$	$169/60$	$-2/5$	$7/240$



Euler Method

$$X'(t) = f(X(t), t)$$

$$X(t_{n+1}) = X(t_n) + \Delta t f(X(t_n), t_n)$$



- 1st order explicit method

Runge–Kutta methods

RK2

$$X(t_{n+1/2}) = X(t_n) + \frac{1}{2}\Delta t f(X(t_n), t_n)$$

$$X(t_{n+1}) = X(t_n) + \Delta t f(X(t_{n+1/2}), t_{n+1/2})$$

- 2nd order explicit method

Trapezoidal Rule

$$X(t_{n+1}) = X(t_n) + \Delta t \left(\frac{1}{2} f(X(t_n), t_n) + \frac{1}{2} f(X(t_{n+1}), t_{n+1}) \right)$$

- 2nd order implicit method

Finite Difference Methods

- Accuracy & truncation error
- Stability & convergence
- Computational efficiency
- Memory efficiency

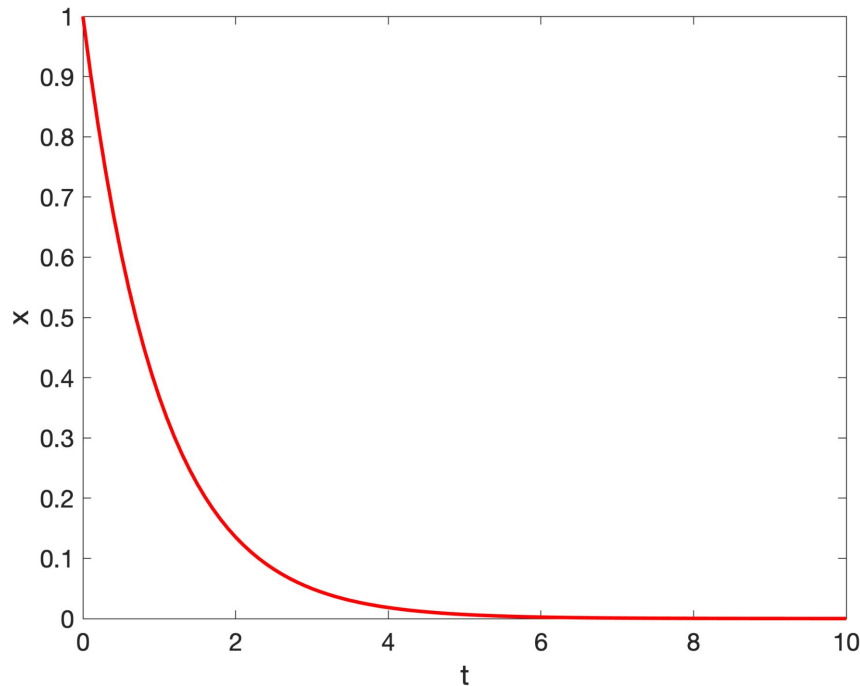
A-Stability

test equation

$$x' = kx$$

$$x = e^{kt}$$

$x \rightarrow 0$ as $t \rightarrow \infty$ if $\Re(k) < 0$



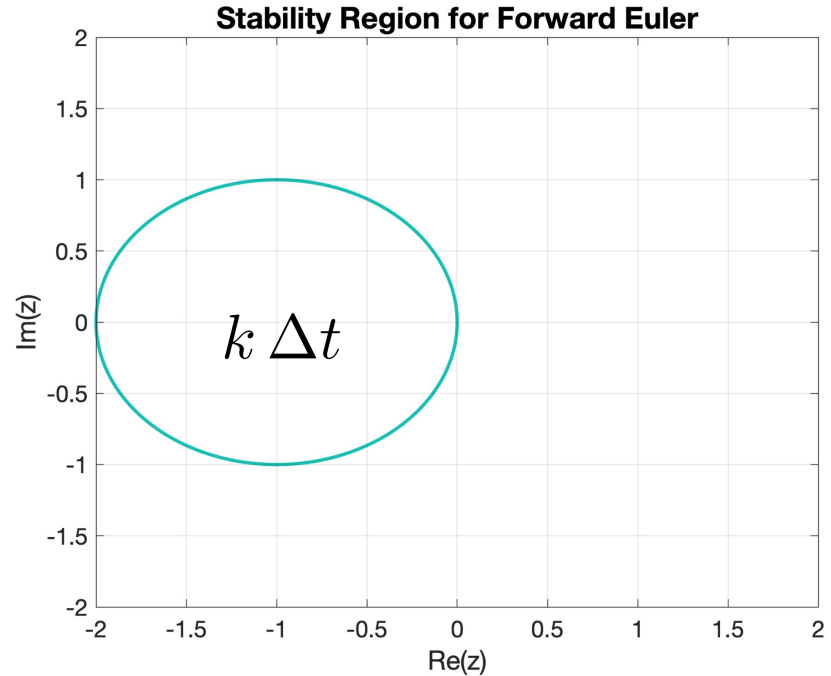
A-Stability

Forward Euler

$$x(t_{n+1}) = x(t_n) + k x(t_n) \Delta t$$

$$x(t_{n+1}) = (1 + k \Delta t) x(t_n)$$

$$x(t_n) = \underbrace{(1 + k \Delta t)^n}_{\text{stable when } < 1} x(t_0)$$



Higher Order ODEs

2nd order ODE $x''(t) = f(t, x(t), x'(t))$

reparameterization $x'(t) = v(t)$
 $v'(t) = f(t, x(t), v(t))$

Euler method $x(t_{n+1}) = x(t_n) + \Delta t v(t_n)$
 $v(t_{n+1}) = v(t_n) + \Delta t f(t_n, x(t_n), v(t_n))$