Mass-Spring Systems

A Basic Tool for Modeling Deformable Objects

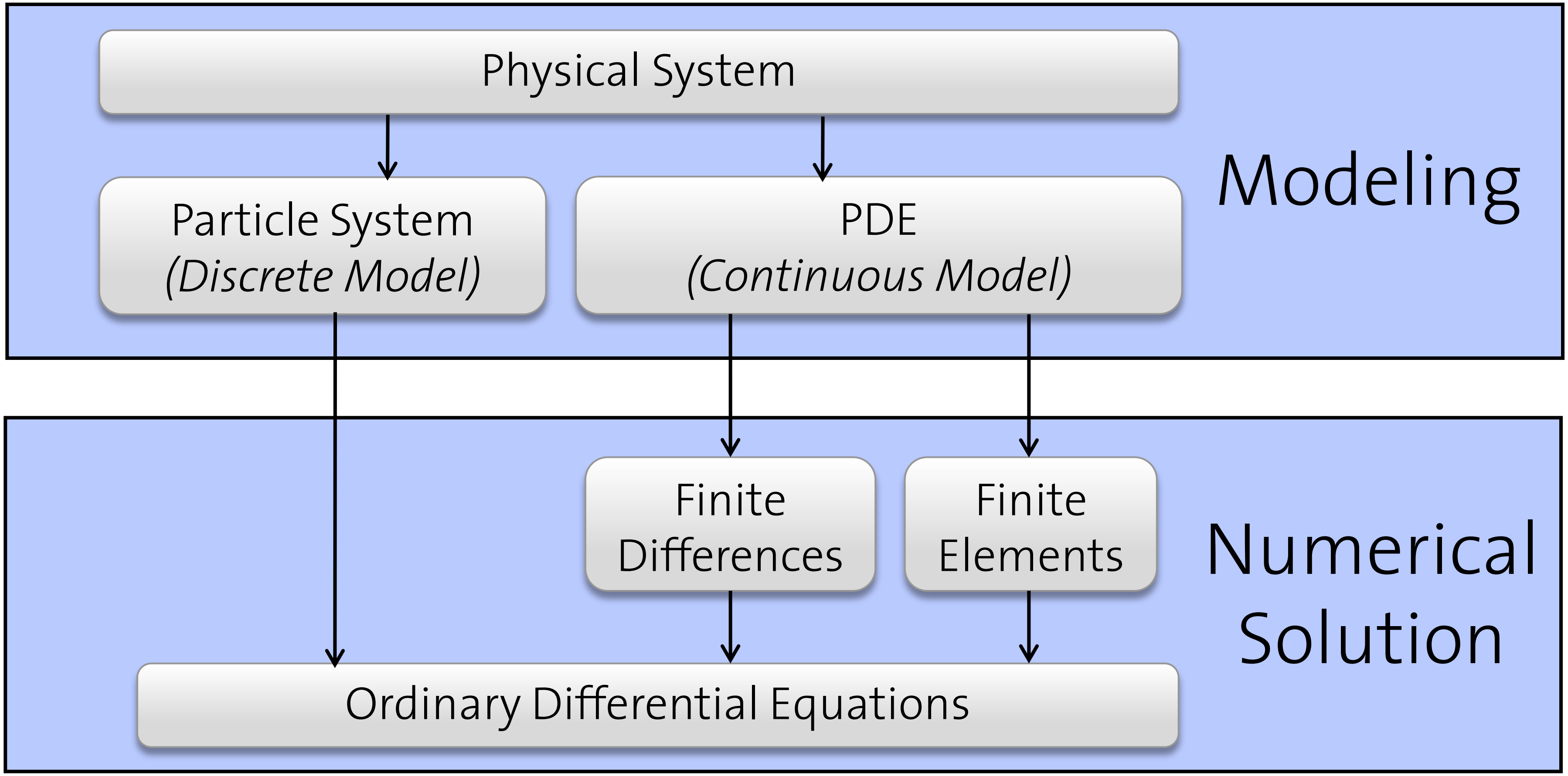
Part 1

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Disney Research

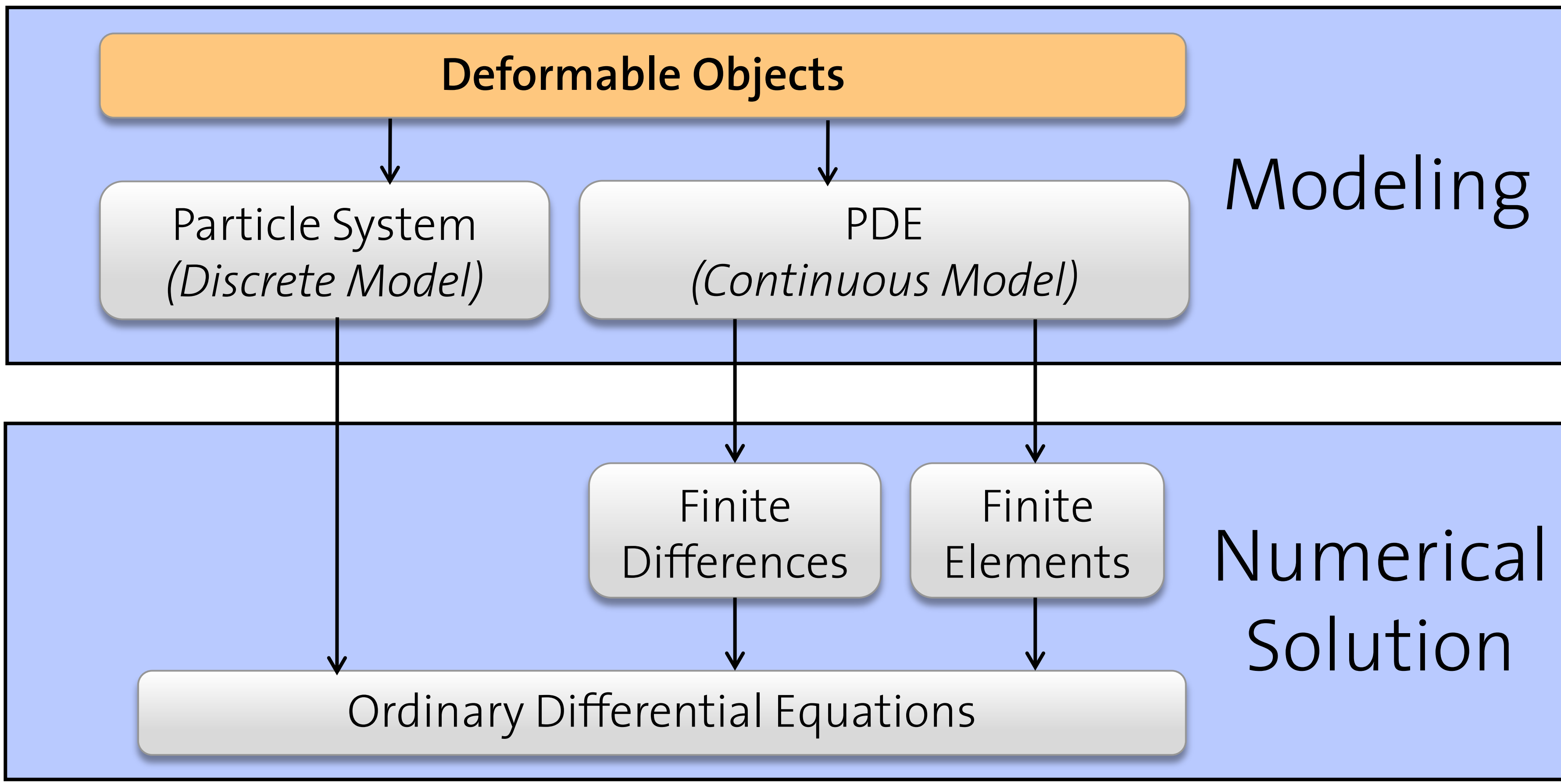
cgl
Computer Graphics Laboratory ETH Zurich

Physical Simulation: How to... ?



Disney Research

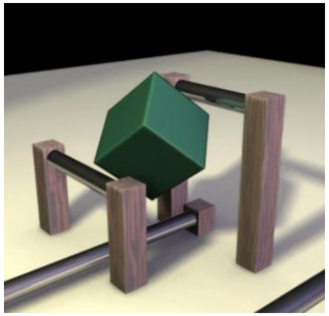
Physical Simulation: How to… ?

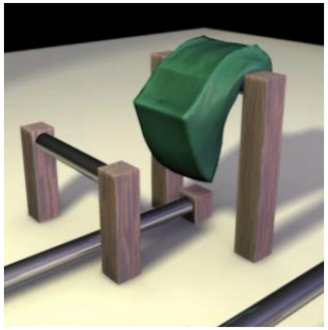


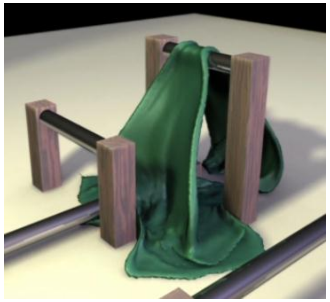
Disney Research

Deformable Objects

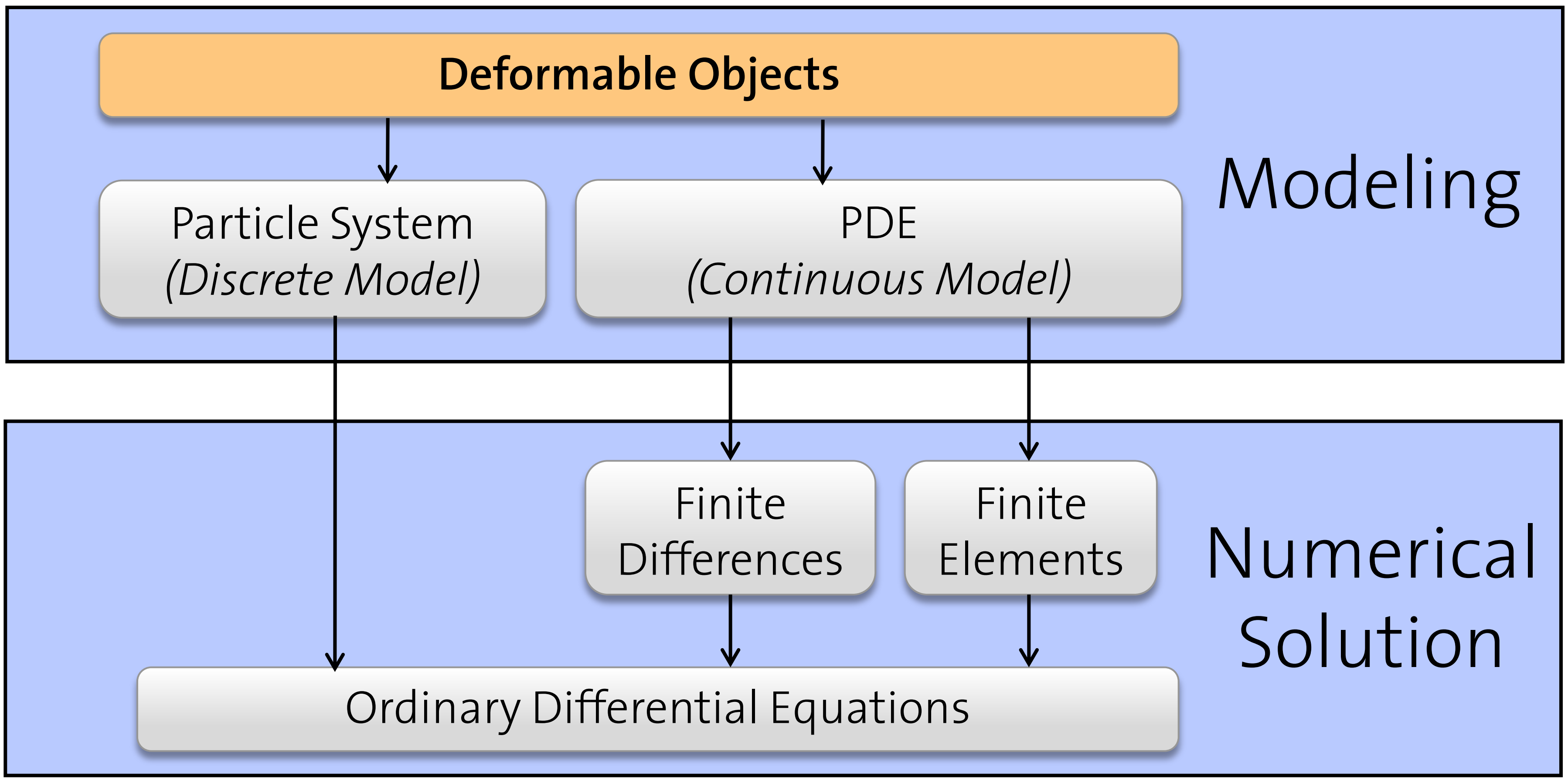
- Deformable objects
  - deformed under applied forces
  - resist deformation
- Common material properties
  - **Elastic**: deformations are reversible
  - **Viscous**: amplitude of oscillations is reduced
  - **Plastic**: irreversible deformations
  - Any combination thereof







Physical Simulation: How to… ?



Disney Research

Physical Simulation: How to… ?

Deformable Objects

Mass-Spring Systems
(Discrete Model)

Modeling

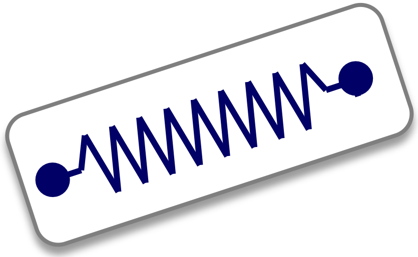
Numerical Solution

Ordinary Differential Equations

Disney Research

Outline

Mass-Spring Systems



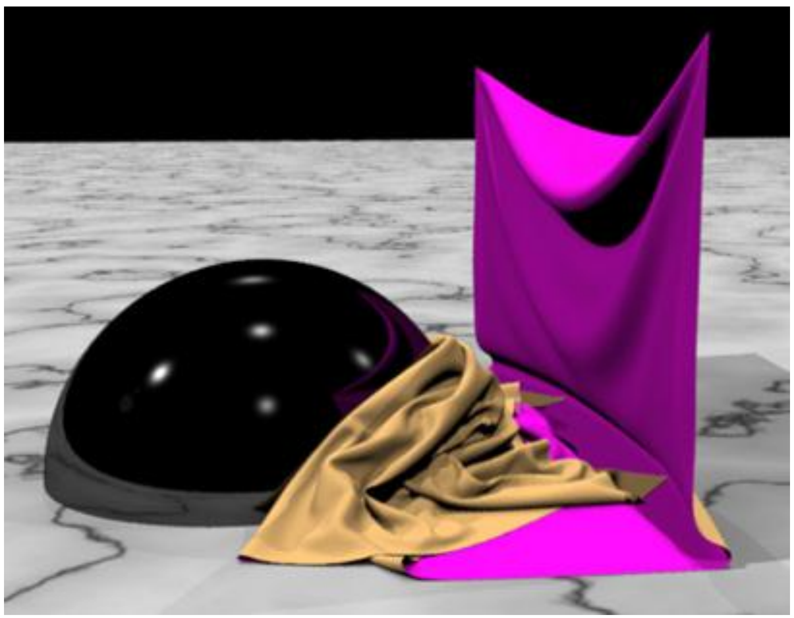
Steps towards simulation

1. **Spatial discretization**: sample object with mass points
2. **Forces**: define internal (*springs!*) and external forces
3. **Dynamics**: set up equations of motion
4. **Temporal discretization**: solve equations of motion

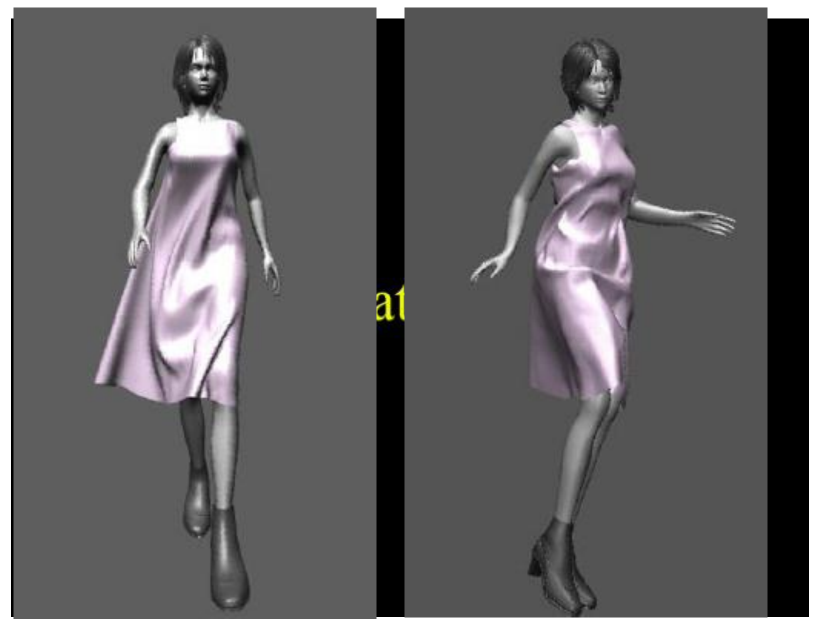
Disney Research

Applications

Cloth Simulation



Bridson et al., 2002



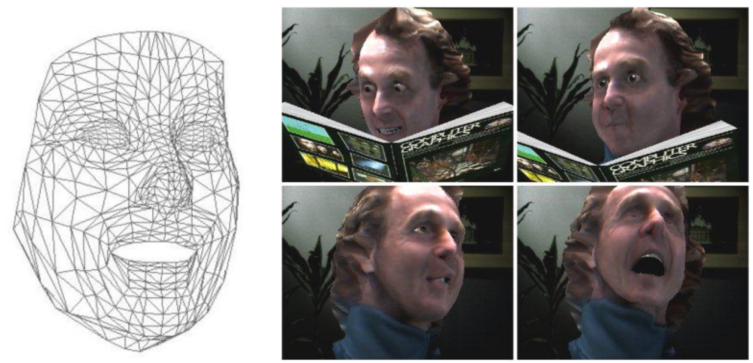
Choi & Ko, 2002

Applications

Hair animation

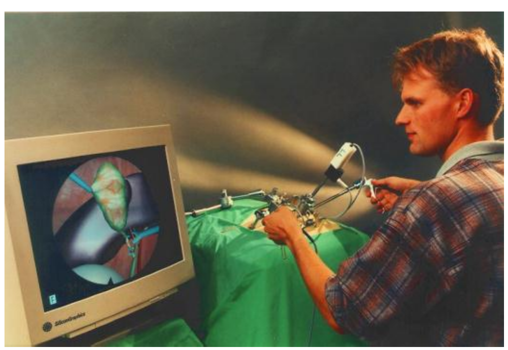
Selle et al., 2008

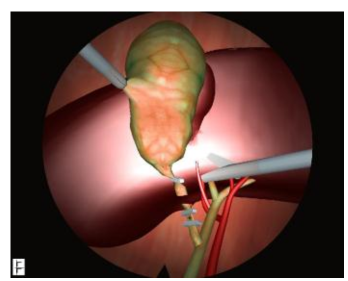
Facial animation



Lee et al., 1995

Medical simulation

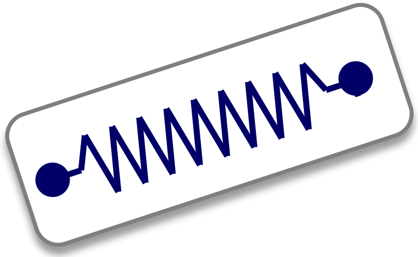




Kuehnapfel et al., 1993

Outline

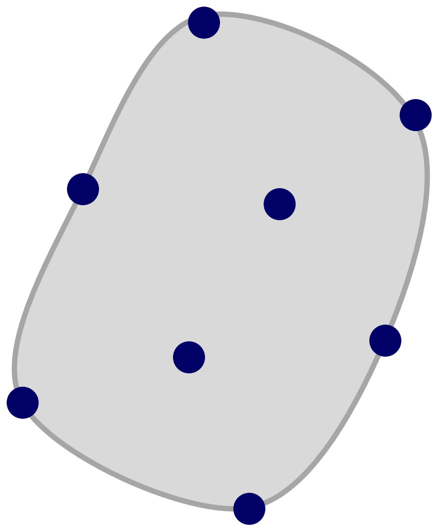
Mass-Spring Systems



Steps towards simulation

1. **Spatial discretization**: sample object with mass points
2. **Forces**: define internal (*springs!*) and external forces
3. **Dynamics**: set up equations of motion
4. **Temporal discretization**: solve equations of motion

Spatial Discretization



Sample object with mass points

- Total mass of object: M
- Number of mass points: n
- Mass of each point: $m = M/n$ (*uniform distribution*)

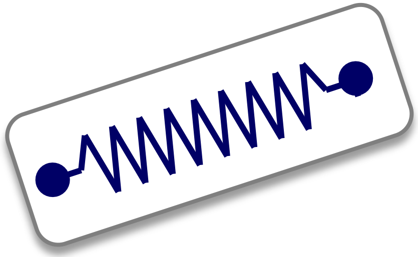
Each point holds properties

- Mass m_i
- Position $\mathbf{x}_i(t)$
- Velocity $\mathbf{v}_i(t)$

Disney Research

Outline

Mass-Spring Systems



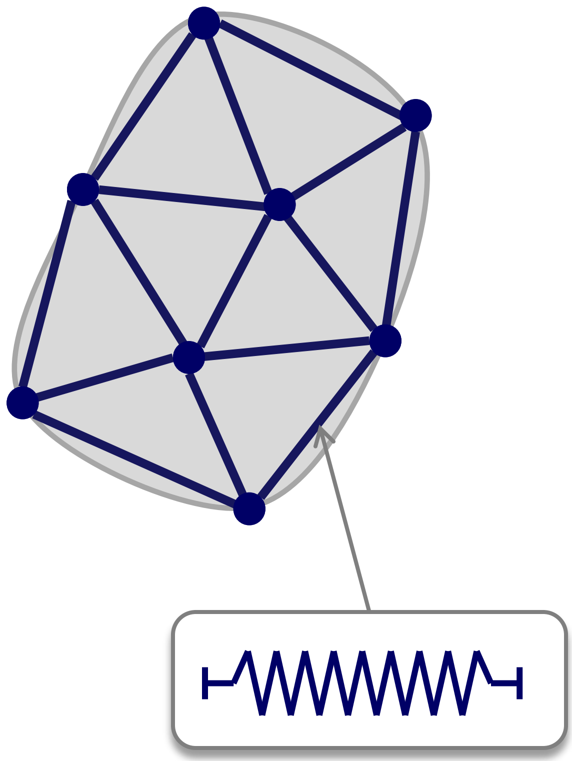
Steps towards simulation

1. **Spatial discretization**: sample object with mass points
2. **Forces**: define internal (*springs!*) and external forces
3. **Dynamics**: set up equations of motion
4. **Temporal discretization**: solve equations of motion

Disney Research

Forces

What are the forces that act on particle i?



External forces

- Gravity $\mathbf{F}_i^g = m_i \begin{pmatrix} 0 \\ 0 \\ 9.81 \end{pmatrix} \frac{m}{s^2}$

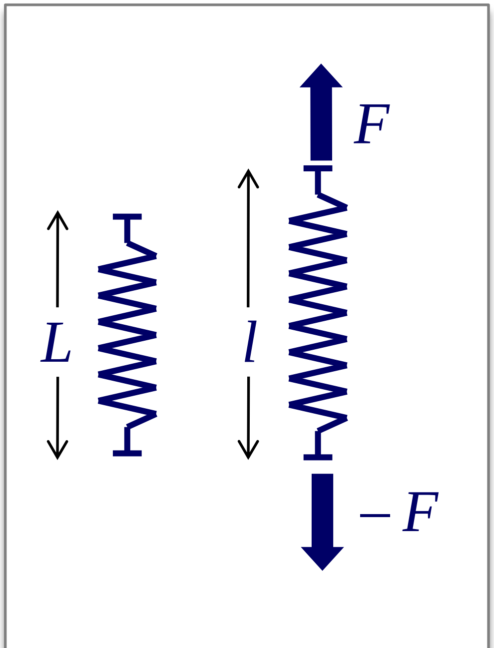
Internal forces

- Elastic spring forces
- Viscous damping forces

Total force $\mathbf{F}_i = \mathbf{F}_i^{\text{int}} + \mathbf{F}_i^{\text{ext}}$

Note: forces are 3D, $\mathbf{F}_i \in \mathbf{R}^3$

Internal Forces: Elastic Springs



Elasticity: *Ability of a spring to return to its initial length when the deforming force is removed.*

Spring Force:

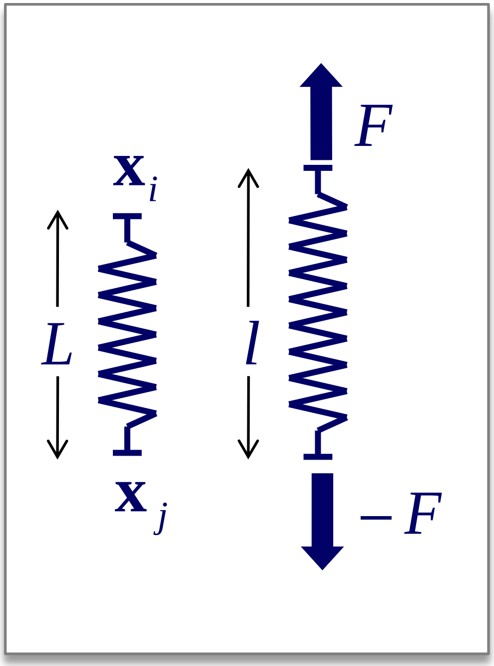
- *Ceiiinosssttuv.* (Hooke, 1676)
- *Ut tensio, sic vis.* (Hooke, 1678)

$\rightarrow$ *Force is linear w.r.t. extension!*

Initial spring length	L
Current spring length	l
Spring stiffness	k

$$F = -k(l - L) \quad \text{Hooke's Law}$$

Internal Forces: Elastic Springs



Force in 1D

$$F = -k\left(l - L\right)$$

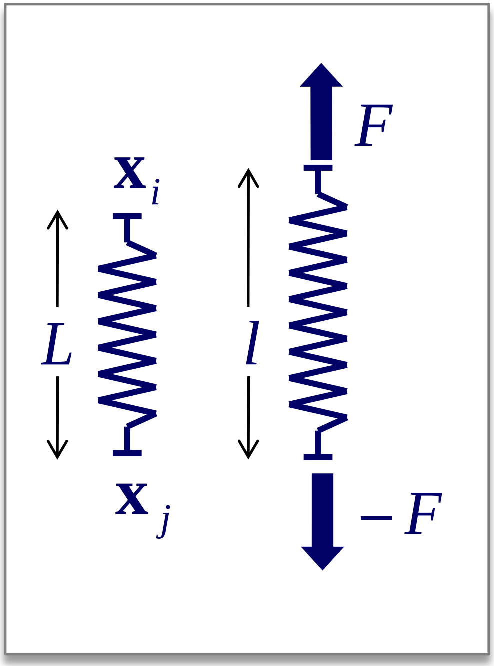
Force in 3D

$$\mathbf{F}_i = -k\left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\| - L\right)\frac{\mathbf{x}_i - \mathbf{x}_j}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}$$

Initial spring length L
Current spring length l
Spring stiffness k

Disney Research

Elastic Energy



For purely elastic springs (*materials*)

- Force depends only on position
- No energy lost during deformation

Work done by forces

$$W = \int_L^l k(x - L)\, dx$$

Elastic spring energy

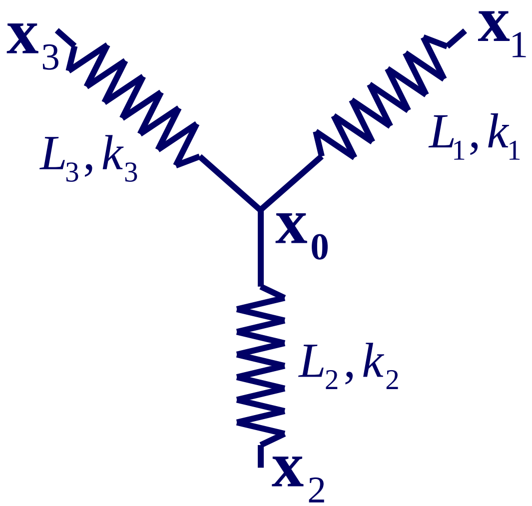
$$E = W = \frac{1}{2} k(l - L)^2$$

Force = – gradient of energy

$$\mathbf{F}_i = -\frac{\partial E}{\partial \mathbf{x}_i}$$

Forces at Mass Point

Internal forces $\mathbf{F}^{\text{int}}$



External forces $\mathbf{F}^{\text{ext}}$

- Gravity
- Contact forces
- All forces that are not caused by springs

Total spring force

$$\mathbf{F}_0^{\text{int}} = - \sum_{i \mid i \in \{1,2,3\}} k_i (l_i - L_i) \frac{\mathbf{x}_i - \mathbf{x}_0}{l_i}$$

Resulting force at point i

$$\mathbf{F}_i = \mathbf{F}_i^{\text{int}} + \mathbf{F}_i^{\text{ext}}$$

Dissipative Forces

- Real-world mechanical systems dissipate energy over time

  Internal friction $\rightarrow$ *Thermal energy (irreversible process)*
- Controllable dissipation useful for physics simulations

  Do we want things to move indefinitely?
- Dissipation for mass-spring systems

Point damping

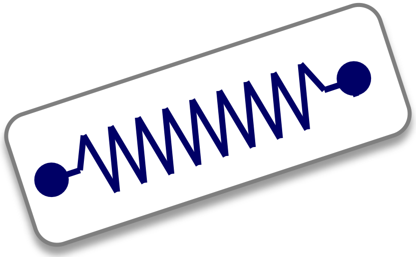
$$\mathbf{F}^{pd}(t) = -\gamma \cdot \mathbf{v}(t)$$

γ is damping coefficient

\+ Simple and efficient
– Damps *all* motion (translations and rotations)

Outline

Mass-Spring Systems



Steps towards simulation

1. **Spatial discretization**: sample object with mass points
2. **Forces**: define internal (*springs!*) and external forces
3. **Dynamics**: set up equations of motion
4. **Temporal discretization**: solve equations of motion

Dynamics

Force is known for every particle.

How do we determine motion $\mathbf{x}_i(t)$ *?*

- Kinematic relations

Velocity

$$\mathbf{v}_i(t) = \frac{d\mathbf{x}_i(t)}{dt}$$

Acceleration

$$\mathbf{a}_i(t) = \frac{d\mathbf{v}_i(t)}{dt} = \frac{d^2\mathbf{x}_i(t)}{dt^2}$$

- Motion follows from

Newton's 2nd Law

$$\mathbf{F}_i = m_i \mathbf{a}_i$$

Equations of Motion

Newton's 2nd Law

$$\mathbf{F}_i = m_i \mathbf{a}_i$$

Acceleration

$$\mathbf{a}_i(t) = \frac{d^2 \mathbf{x}_i(t)}{dt^2}$$

Equations of motion
for one mass point
(*3 equations*)

$$m_i \frac{d^2 \mathbf{x}_i(t)}{dt^2} = \mathbf{F}_i^{\text{int}}(t) + \mathbf{F}_i^{\text{ext}}(t)$$

Equations of motion
for system of mass points
(*3n equations*)

$$\mathbf{M} \frac{d^2 \mathbf{x}(t)}{dt^2} = \mathbf{F}^{\text{int}}(t) + \mathbf{F}^{\text{ext}}(t)$$

$\mathbf{M} \in \mathbf{R}^{3n \times 3n}$ is a diagonal matrix

Equations of Motion

Special case:
point damping

$$\mathbf{F}^{pd}(t) = -\gamma \cdot \mathbf{v}(t)$$

Equations of motion
(3n equations)

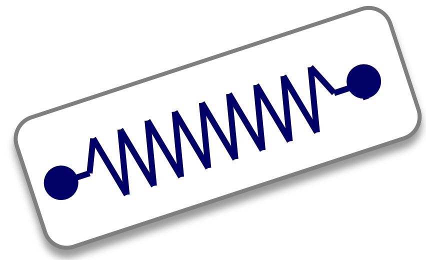
$$\mathbf{M}\frac{d^2\mathbf{x}(t)}{dt^2} + \mathbf{D}\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}^{\mathrm{int}}(t) + \mathbf{F}^{\mathrm{ext}}(t)$$

$\mathbf{D} \in \mathbf{R}^{3n \times 3n}$ is diagonal with entries γ

Motion is determined by a system of 3n
2nd *order Ordinary Differential Equations*

Outline

Mass-Spring Systems



Steps towards simulation

1. **Spatial discretization**: sample object with mass points
2. **Forces**: define internal (*springs!*) and external forces
3. **Dynamics**: set up equations of motion
4. **Temporal discretization**: solve equations of motion

Differential Equations

- A **differential equation** describes an unknown function through its derivatives
- An ordinary differential equation (ODE) contains only derivatives with respect to a single variable

Example

$$\mathbf{x}''(t) = \frac{\mathbf{F}(t) - \gamma\,\mathbf{x}'(t)}{m_i}$$

$\Rightarrow$

Abstract form

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

- Expressed as function of highest derivative

$$y^{(n)} = f(x, y, y', y'', \ldots, y^{(n-1)})$$

n is the order of the ODE

Initial Value Problems

- Solving an ODE: given f, determine y

Example

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

$\Rightarrow$

Solution

$$y(t) = Ce^{t}$$

- Integration constant C is unknown, determined by

Initial value $y(0) = 2$ $\Rightarrow$ $C = 2$

- ODE + initial value = *initial value problem* (IVP)

Picard–Lindelöf theorem:

IVP has unique solution if f is Lipschitz continuous!

Mass-Spring Systems

Applied to mass-spring systems?

We have:

- Initial position $\mathbf{x}_i(t_0)$
- Initial velocity $\mathbf{v}_i(t_0)$
- Governing ODE *(2nd order)*

$$\boxed{\frac{d^2\mathbf{x_i}(t)}{dt^2} = \frac{\mathbf{F_i}(t) - \gamma \mathbf{v_i}(t)}{m_i}}$$

We want:

- position $\mathbf{x}_i(t)$ over time

Disney Research

Rewriting the Problem

- *Easier* to deal with first order ODE
- Reduce 2nd order ODE to two coupled 1st order ODEs

$$m_i \frac{d^2\mathbf{x}_\mathbf{i}(t)}{dt^2} + \gamma \frac{d\mathbf{x}_\mathbf{i}(t)}{dt} = \mathbf{F}_i(t)$$

$$\Downarrow$$

velocity

acceleration

$$\frac{d\mathbf{x}_\mathbf{i}(t)}{dt} = \mathbf{v}_\mathbf{i}(t) \qquad \frac{d\mathbf{v}_\mathbf{i}(t)}{dt} = \frac{\mathbf{F}_\mathbf{i}(t) - \gamma\mathbf{v}_\mathbf{i}(t)}{m_i}$$

Disney Research

Rewriting the Problem

- Two coupled 1st order ODEs (2 times 3 equations)

$$\frac{d\mathbf{x_i}(t)}{dt} = \mathbf{v_i}(t) \qquad \frac{d\mathbf{v_i}(t)}{dt} = \frac{\mathbf{F_i}(t) - \gamma \mathbf{v_i}(t)}{m_i}$$

- Write as one system of 1st order ODEs

$$\mathbf{y}_i(t) = \begin{pmatrix} \mathbf{x}_i(t) \\ \mathbf{v}_i(t) \end{pmatrix} \qquad \mathbf{y}_i'(t) = \begin{pmatrix} \mathbf{v}_i(t) \\ \dfrac{\mathbf{F}_i(t) - \gamma \mathbf{v}_i(t)}{m_i} \end{pmatrix}$$

Solution

Given first order ODE with initial conditions, how can we solve for $y(t)$*?*

$$y'(t) = f(t, y)$$
$$y(t_0) = y_0$$

Analytical solution

- Provides exact solution *y(t)* at any time value *t*
- In general not available (impractical for complex systems)

(t_0, y_0)

Disney Research

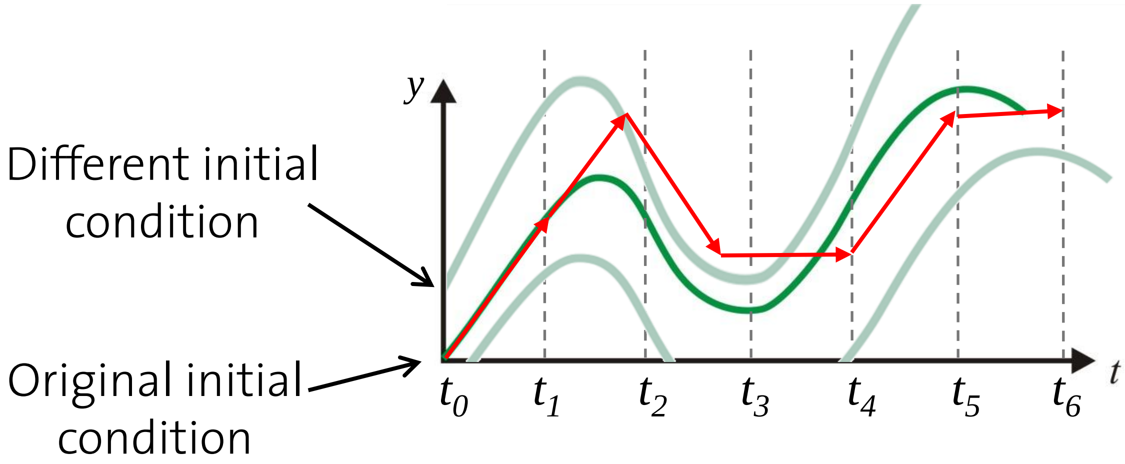
Solution

Given first order ODE with initial conditions, how can we solve for $y(t)$*?*

$$y'(t) = f(t, y)$$
$$y(t_0) = y_0$$

Numerical solution

- Compute approximations y_i to true solution for **discrete** time instants t_i
- Compute solution at t_{i+1} based on previous solutions at t_{i-1}, t_{i-2},...



$$y_n \rightarrow y_{n+1}$$

Numerical Integration

- Solve ODEs numerically ⇨ *Numerical Integration*
- Why called like that? $y(t+h) = y(t) + \int_t^{t+h} y'(t)\, dt$
- Integration schemes

Explicit Methods
Explicit Euler
Heun, Midpoint
Runge-Kutta methods

Implicit Methods
Backward Euler
Implicit Midpoint
BDF methods

Methods for higher order ODEs
Verlet
Leapfrog
Newmark methods

Computing Approximations

- Notation: $y(t)$ true solution
  y_i approximate solutions at $t_i = t_0 + i \cdot h$
  h time step (*fix*)
- Problem definition: given y_n, compute y_{n+1}
- How do we get from *y(t)* to *y(t+h)*?

Taylor expansion $$y(t+h) = y(t) + \frac{y'(t)}{1!} h + \frac{y''(t)}{2!} h^2 + \ldots$$

- Compute first order approximation

$$y(t+h) \approx y(t) + h \cdot y'(t) \longrightarrow y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

Euler's Method

Euler Step (1768)

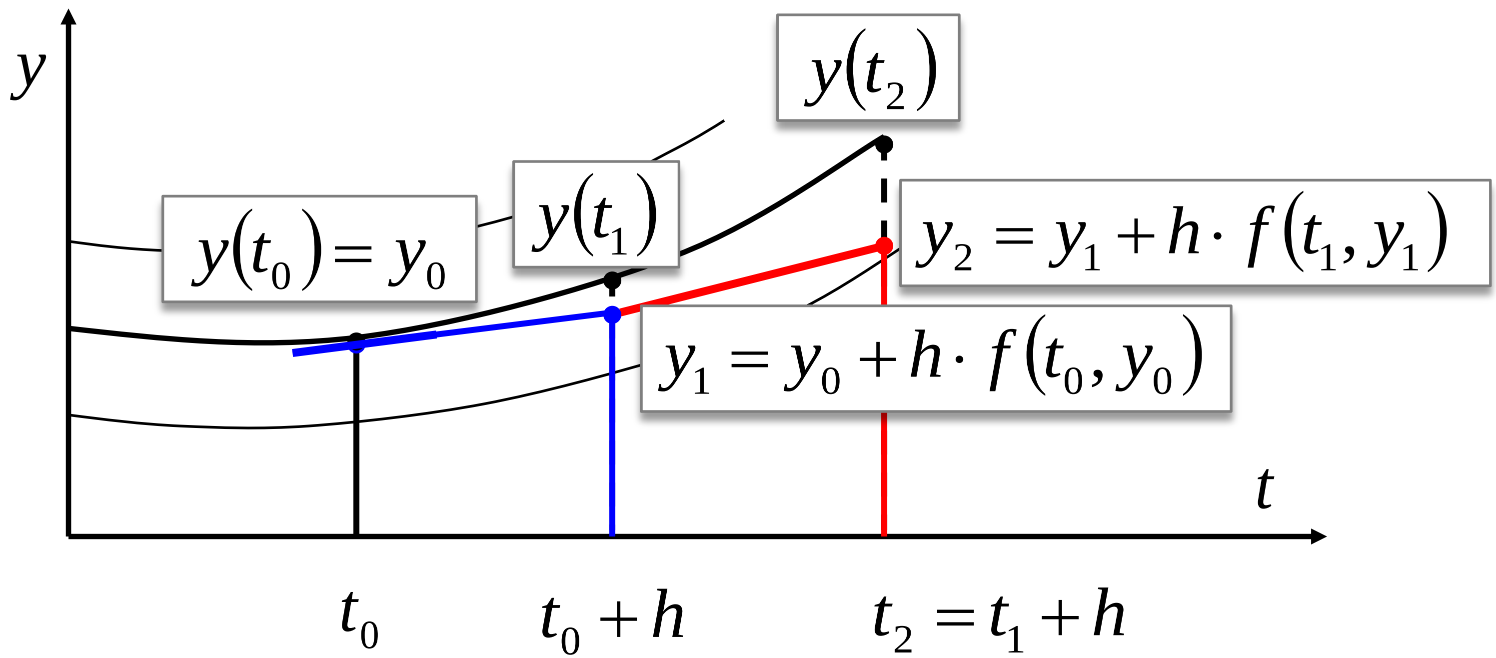
$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

- Idea: start at initial condition and take a step into the direction of the tangent.
- Iteration scheme: $y_n \rightarrow f(t_n, y_n) \rightarrow y_{n+1} \rightarrow f(t_{n+1}, y_{n+1}) \rightarrow \ldots$
- Bear in mind: *computations are approximate!*

$$y_1 = y_0 + h \cdot f(t_0, y_0) \qquad \textit{but} \qquad y_2 = y_1 + h \cdot f(t_1, y_1)$$

$$f(t_0, y_0) = f(t_0, y(t_0)) \qquad\qquad f(t_1, y_1) \neq f(t_1, y(t_1))$$

Euler's Method Graphically



Disney Research

Euler's Method

For 2nd order ODE

$$\mathbf{x}'(t) = \mathbf{v}(t) \qquad \mathbf{v}'(t) = \frac{\mathbf{F}(t) - \gamma \mathbf{v}(t)}{m}$$

Compute	$\mathbf{x}(t_0 + h) = \mathbf{x}(t_0) + h\mathbf{x}'(t_0) = \mathbf{x}(t_0) + h\mathbf{v}(t_0)$
Compute	$\mathbf{v}(t_0 + h) = \mathbf{v}(t_0) + h\mathbf{v}'(t_0) = \mathbf{v}(t_0) + h\dfrac{\mathbf{F}(t_0) - \gamma \mathbf{v}(t_0)}{m}$

F*(t)* is computed from **x***(t)* and external forces!

Analysis

How to evaluate an integration scheme?

Criteria

- **Convergence**: do approximations converge to true solution, i.e. $h \to 0 \quad \Rightarrow \quad y_i \to y(t_i)$?
- **Accuracy**: how fast does the error decrease as $h \to 0$?
- **Stability**: is the solution always bounded, i.e. $|y_n| < \infty$?
- **Efficiency**: is a given method a *good choice* for a given problem?

Disney Research

Accuracy

- Numerical solution exhibits error

$$|[y_n + \int_{t_n}^{t_{n+1}} y'(t)\,dt] - y_{n+1}|$$

Local error (*single step*)

$$|\, y_i - y(t_i)\,|$$

Global error (*accumulated*)

- Accuracy of integration schemes:

  Local error is $O(h^{p+1}) \rightarrow$ *method is accurate of order p!*

- Error depends on the step size:

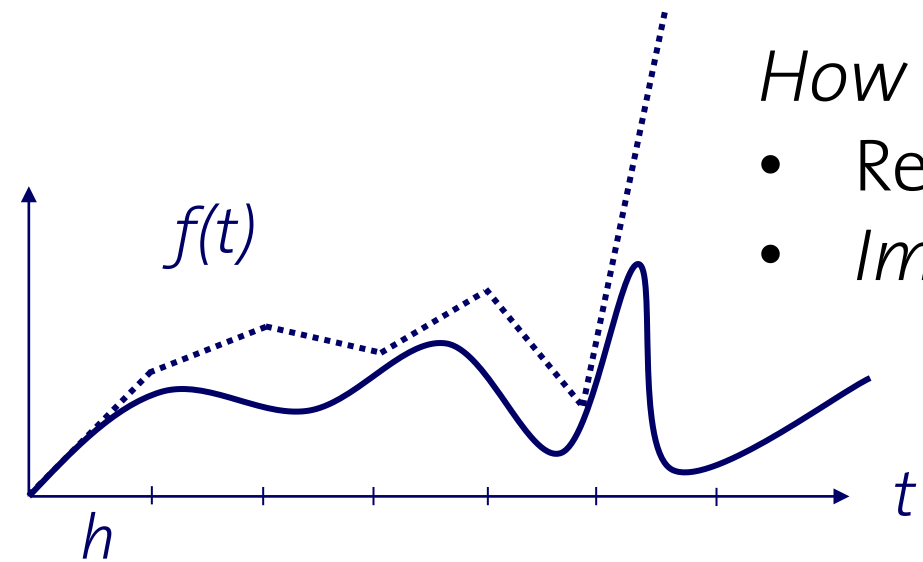
  Explicit Euler is order 1 $\rightarrow$ $O(h^2)$ *error per step!*

Problems

- Numerical integration is inaccurate
- Error accumulates
- can cause instability

$$\underbrace{f(t+h) = f(t) + f'(t)h}_{\text{Euler step}} + \underbrace{O(h^2)}_{\text{Error}}$$

Error $0 \le e < \frac{h^2}{2} \cdot f''(t_e), \quad t_e \in [t, t+h]$



How can we reduce the error?

- Reduce step size
- *Improve accuracy*

Disney Research

Higher Accuracy

Taylor expansion $$y(t+h) = y(t) + \frac{y'(t)}{1!}h + \frac{y''(t)}{2!}h^2 + \ldots$$

- Idea: use second order approximation (i.e. O(h^3))

$$y(t+h) \approx y(t) + h \cdot y'(t) + \frac{h^2}{2} y''(t) \tag{1}$$

- $y''(t)$ unknown → use difference approximation

$$y''(t) = \frac{y'(t+h) - y'(t)}{h} + O(h) \tag{2}$$

- Use (2) in (1): $$y(t+h) \approx y(t) + \frac{h}{2}[y'(t) + y'(t+h)] \tag{3}$$

- Eq. (3) is O(h^3) as long as Eq. (2) is O(h)

Higher Accuracy

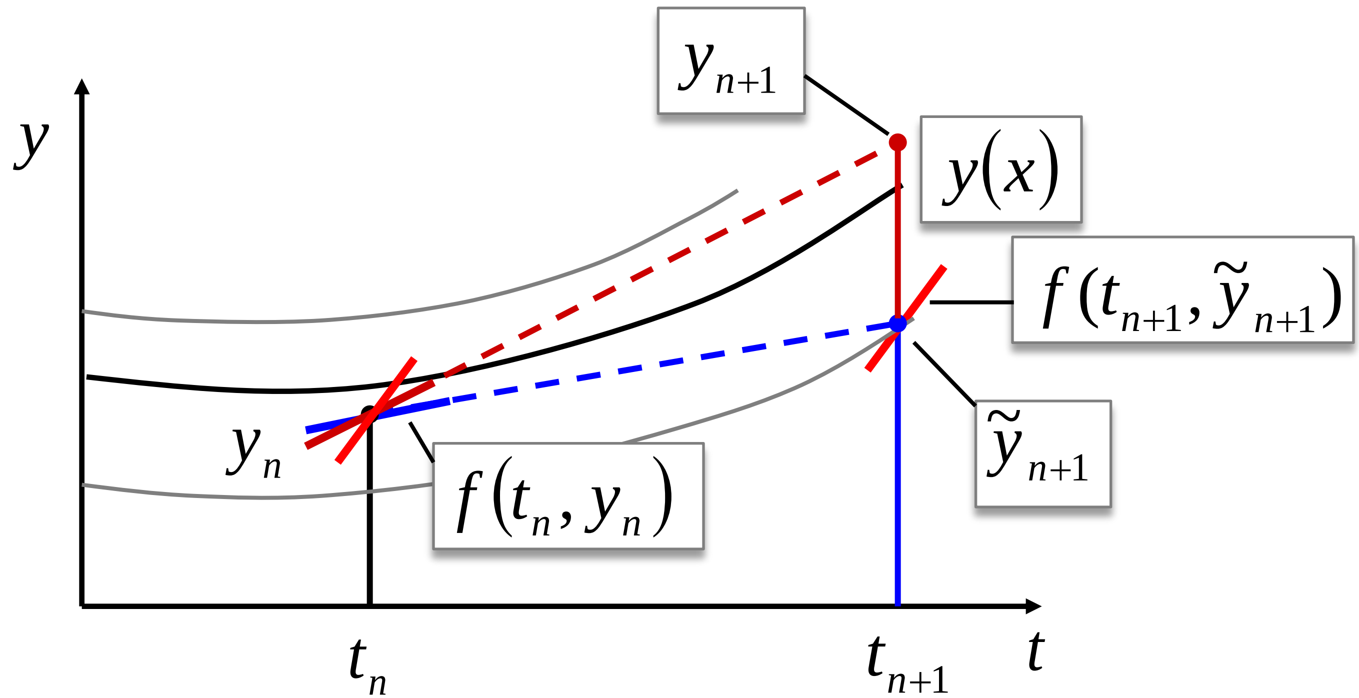
$$y(t+h) \approx y(t) + \frac{h}{2}[y'(t) + y'(t+h)] \qquad (3)$$

- $y'(t+h)$ unknown $\rightarrow$ use Euler step
  - Step $\tilde{y}_{n+1} = y_n + h \cdot f(t_n, y_n)$
  - Evaluate $y'_{n+1} = f(t_{n+1}, \tilde{y}_{n+1})$ (4)
- Putting it together: **Heun's method**

$$\boxed{\begin{aligned} \tilde{y}_{n+1} &= y_n + h \cdot f(t_n, y_n) \\ y_{n+1} &= y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, \tilde{y}_{n+1})] \end{aligned}}$$

2^{nd} order accurate!

Heun's Method Graphically



Disney Research

Heun's Method

For 2nd order ODE

$$\mathbf{x}'(t) = \mathbf{v}(t) \qquad \mathbf{v}'(t) = \mathbf{a}(\mathbf{x}(t), \mathbf{v}(t)) = \frac{\mathbf{F}(t) - \gamma \mathbf{v}(t)}{m}$$

Compute *v* at *t*	$\mathbf{k}_1 = \mathbf{v}(t)$
Compute *a* at *t*	$\mathbf{l}_1 = \mathbf{a}(\mathbf{x}(t), \mathbf{v}(t))$
Compute *v* at *t+h*	$\mathbf{k}_2 = \mathbf{v}(t) + \mathbf{l}_1 h$
Compute *a* at *t+h* with *x* and *v* at *t+h*	$\mathbf{l}_2 = \mathbf{a}(\mathbf{x}(t) + \mathbf{k}_1 h, \mathbf{k}_2)$
Compute *x* at *t+h* with its velocity at *t* and *t+h*	$\mathbf{x}(t+h) = \mathbf{x}(t) + h\dfrac{\mathbf{k}_1 + \mathbf{k}_2}{2}$
Compute *v* at *t+h* with the acceleration at *t,t+h*	$\mathbf{v}(t+h) = \mathbf{v}(t) + h\dfrac{\mathbf{l}_1 + \mathbf{l}_2}{2}$

Explicit Midpoint Method

- Heun's method uses *f(t)* and *f(t+h)* to achieve 2nd order accuracy
- Alternative: use *f(t+h/2)* $\rightarrow$ **Midpoint method**

$$y'(t+\tfrac{h}{2}) = \frac{y(t+h)-y(t)}{h} + O(h^2)$$ $\rightarrow$ 2nd order accuracy!

$$\tilde{y}_{n+1/2} = y_n + \tfrac{h}{2} \cdot f(t_n, y_n)$$
$$y_{n+1} = y_n + h \cdot f(t_{n+1/2}, \tilde{y}_{n+1/2})$$

- Note: both Heun and Midpoint achieve 2nd order via two evaluations of *f*

Disney Research

4^{th}-Order Runge-Kutta Method

- RK4 is one of the most widely used integrators
- Four slope evaluations $\rightarrow$ 4^{th}-order accuracy:

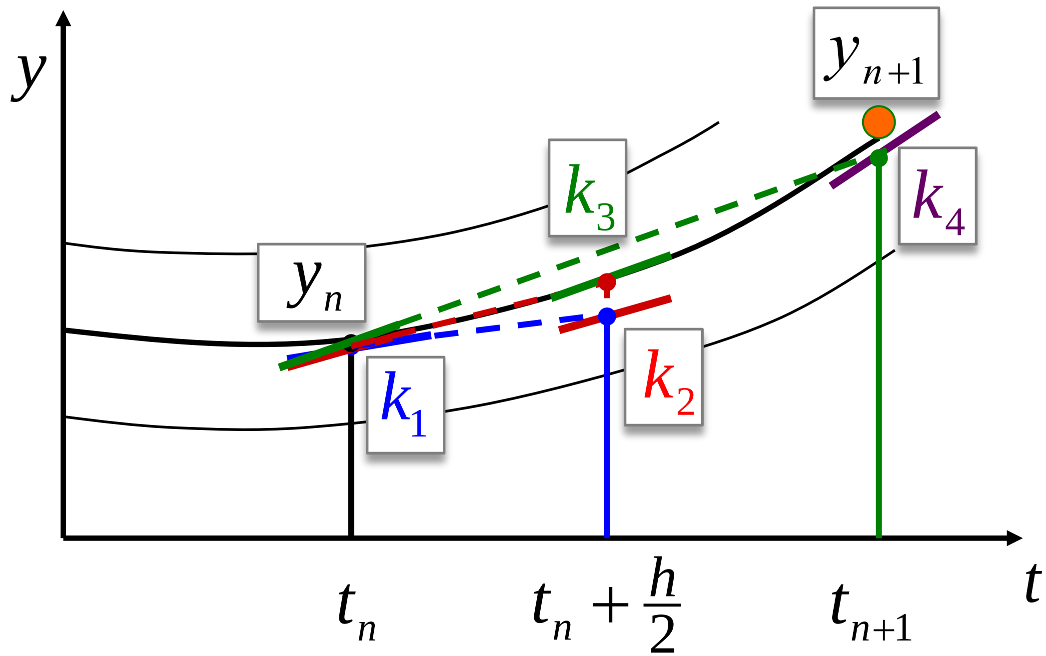
$k_1 = f(t_n, y_n)$	slope at beginning of interval
$k_2 = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1)$	slope at mid-interval
$k_3 = f(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2)$	corrected slope at mid-interval
$k_4 = f(t_n + h, y_n + hk_3)$	slope at end of interval

- Use weighted average slope

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

Disney Research

RK4 Graphically



$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

Disney Research

4th-Order Runge-Kutta

For 2nd order ODE

$$\mathbf{k}_1 = \mathbf{v}(t)$$

$$\mathbf{l}_1 = \mathbf{a}(\mathbf{x}(t), \mathbf{v}(t))$$

$$\mathbf{k}_2 = \mathbf{v}(t) + \mathbf{l}_1 \frac{h}{2}$$

$$\mathbf{l}_2 = \mathbf{a}\left(\mathbf{x}(t) + \mathbf{k}_1 \frac{h}{2}, \mathbf{k}_2 \right)$$

$$\mathbf{k}_3 = \mathbf{v}(t) + \mathbf{l}_2 \frac{h}{2}$$

$$\mathbf{l}_3 = \mathbf{a}\left(\mathbf{x}(t) + \mathbf{k}_2 \frac{h}{2}, \mathbf{k}_3 \right)$$

$$\mathbf{k}_4 = \mathbf{v}(t) + \mathbf{l}_3 h$$

$$\mathbf{l}_4 = \mathbf{a}(\mathbf{x}(t) + \mathbf{k}_3 h, \mathbf{k}_4)$$

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

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

Comparison

Which method is the best?

Comparing integration schemes is difficult

- Different costs per step
- Depends strongly on problem
- Not a single *best* method

Tools from numerical mathematics

- Evaluate on (standard) model problems
- Use work-precision diagrams

Disney Research

Test Equation

Dahlquist's Equation

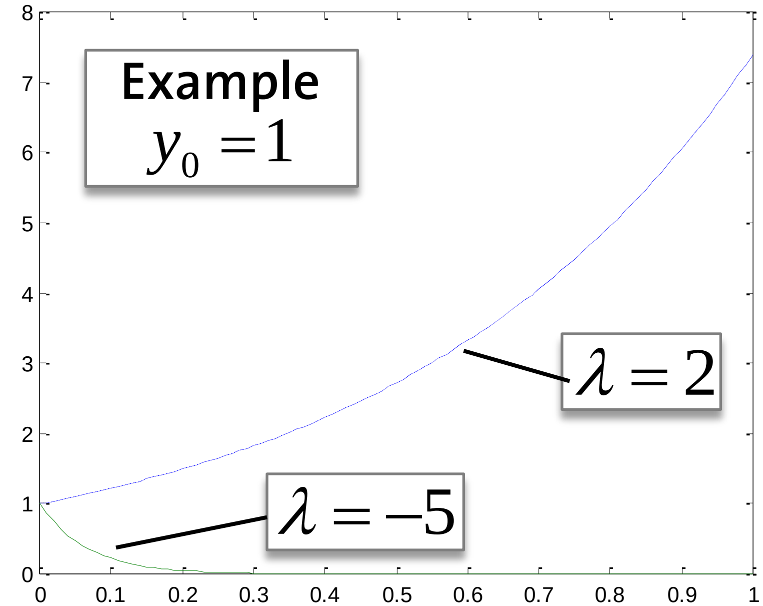
$$y'(t) = \lambda y(t)$$

Analytical Solution

$$y(t) = e^{\lambda t} y_0$$

For $\lambda \in \mathrm{R}$

- $\lambda > 0$ exponential growth
- $\lambda < 0$ exponential decay



Test Equation

Dahlquist's Equation

$$y'(t) = \lambda y(t)$$

Analytical Solution

$$y(t) = e^{\lambda t} y_0$$

For $\lambda \in \mathbb{C}:$ $\lambda = a + ib$ $\Rightarrow$ $y(t) = y_0 \cdot \boxed{e^{at}} \cdot \boxed{e^{ibt}}$

damping

oscillation

- $a < 0$ damped oscillator
- $a = 0$ undamped oscillator
- $a > 0$ unstable

Prototype of mass-spring systems

Euler's Formula

$$e^{ibt} = \cos(tb) + i\sin(tb)$$

Numerical Example

Undamped Harmonic Oscillator

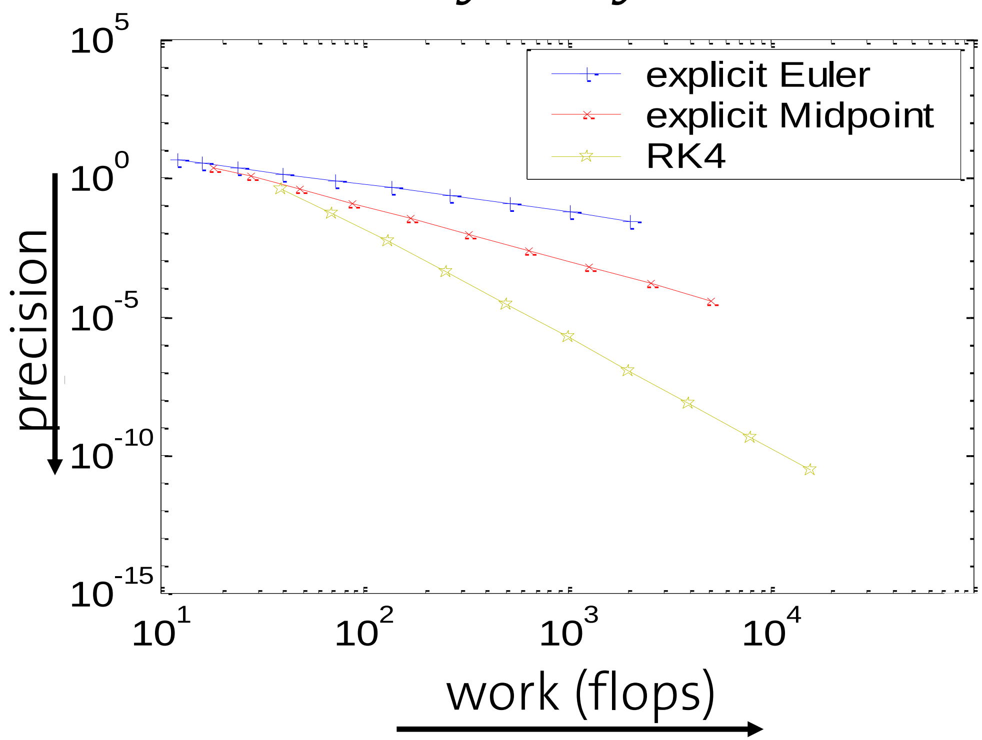
Maplesoft™

Maple 16 is available (for free) at ides.ethz.ch

Explicit Methods

Test equation $y' = \lambda y \quad (\lambda \in \mathbb{R})$

$$y' = 2y$$



Work precision diagram

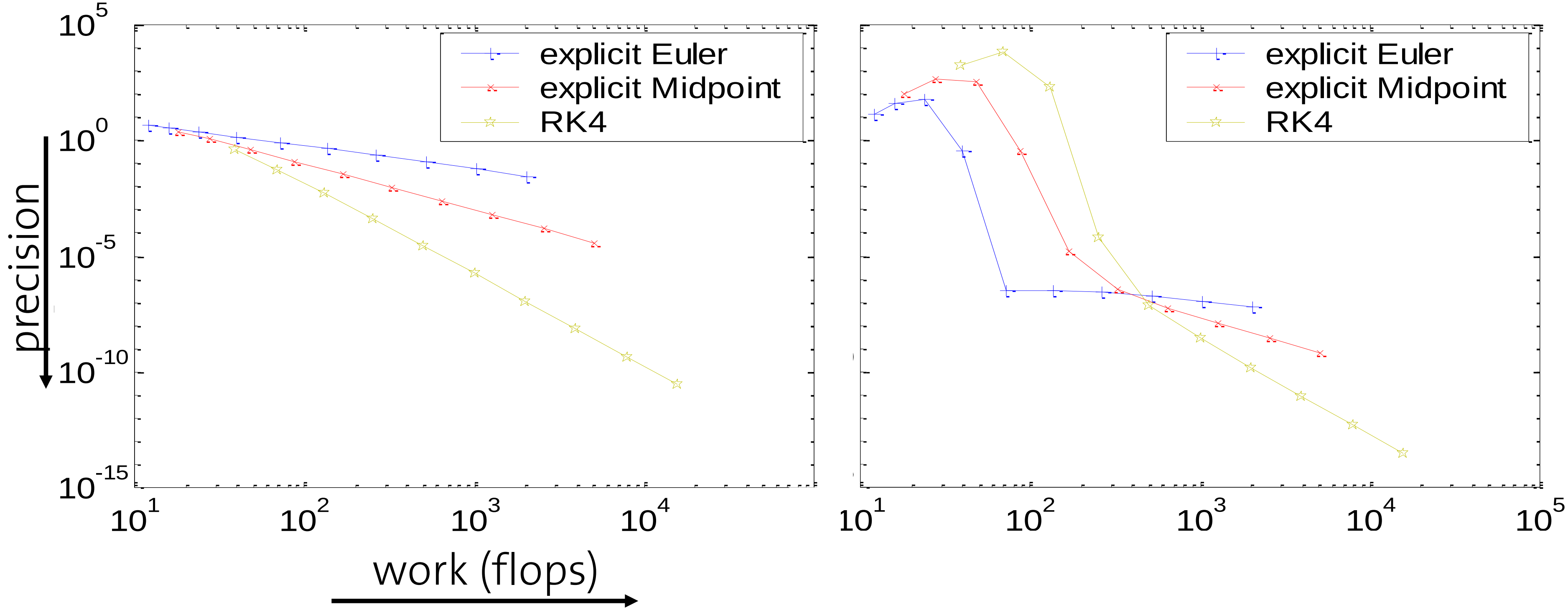
- precision = $| y_{end} - y(t_{end}) |$
- work = flops
- *log-log plot*: functions x^b are lines with slope b
- slope $\rightarrow$ order of method

Disney Research

Explicit Methods

Test equation $y' = \lambda y \quad (\lambda \in \mathrm{R})$

$y' = 2y$ $\qquad$ $y' = -5y$?



Numerical Example

Damped Harmonic Oscillator

Maplesoft™

Stability Problem

What went wrong?

- Analyze
  - Test equation $y' = \lambda y, \quad \lambda < 0$
  - Explicit Euler $y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda) y_n$
- Solve recursion $y_{n+1} = (1 + h\lambda)^{n+1} y_0$

$\Rightarrow$ **Step size restriction** (*explicit Euler*)

$$y_{n+1} < \infty \quad \Leftrightarrow \quad |1 + h\lambda| < 1$$

Stiff Problems

Observations from test equation: explicit methods

- require very small time steps for stable integration
- are inefficient since step size is determined by stability, not accuracy requirements

Problems with this characteristic are termed *stiff*

Don't use explicit methods for stiff problems,
use *implicit* methods.