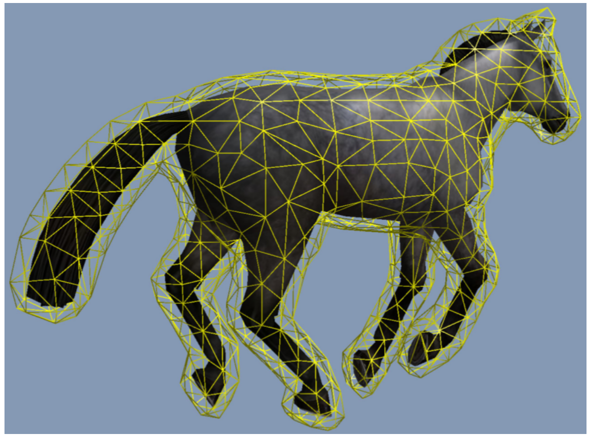
Virtual Reality & Physically-Based Simulation
Mass-Spring-Systems



G. Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de

Newton's Laws

1. Law (law of inertia):

> A body, which no forces act upon, continues to move with constant velocity.

- A resting body is just a special case of this law.

2. Law (law of action):

> If a force **F** acts on a body with mass *m* , then the body accelerates, and its acceleration is given by
>
> $$\mathbf{F} = m \cdot \mathbf{a}$$

- In other words: force and acceleration are proportional to each other; (the proportionality factor happens to be *m)*. In aprticular, both force and acceleration have the same direction.

3. Law (law of reaction):

> If a force **F**, that acts on a body, is extended to another body,
> Then the opposite force –**F** acts on that other body.

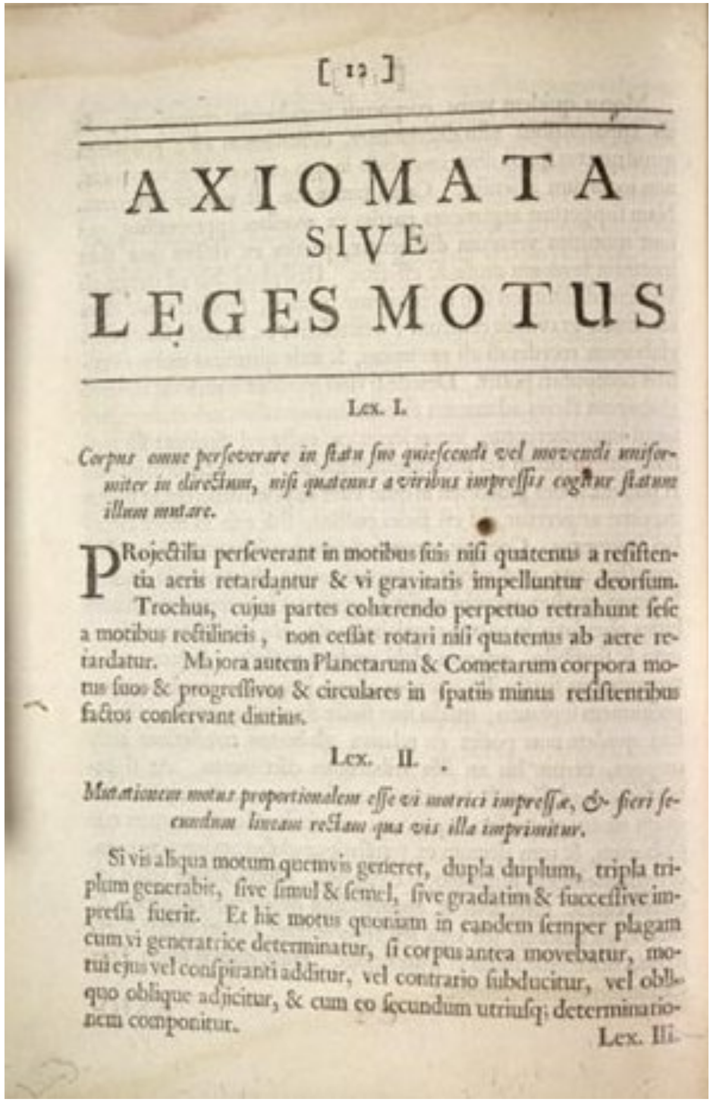
- In school, you learn: "action= reaction"

4. Law (law of superposition):

> If a number offorces $\mathbf{F}_1, \ldots, \mathbf{F}_n$ act on a point or body, then they can be accumulated by vector addition yielding one resulting force:
>
> $$\mathbf{F} = \mathbf{F}_1 + \ldots + \mathbf{F}_n \ .$$

Historical Digression

- Newton published these laws in his original book
  Principia Mathematica
  (1687):
  - *Lex I. Corpus omne perseverare in statu suo quiescendi vel movendi uniformiter in directum, nisi quatenus illud a viribus impressis cogitur statum suum mutare.*
  - *Lex II. Mutationem motus proportionalem esse vi motrici impressae, et fieri secundum lineam rectam qua vis illa imprimitur.*


[12]

AXIOMATA
SIVE
LEGES MOTUS

Lex. I.

Corpus omne perſeverare in ſtatu ſuo quieſcendi vel movendi uniformiter in directum, niſi quatenus a viribus impreſſis cogitur ſtatum illum mutare.

PRojectilia perſeverant in motibus ſuis niſi quatenus a reſiſtentia aeris retardantur & vi gravitatis impelluntur deorſum. Trochus, cujus partes cohærendo perpetuo retrahunt ſeſe a motibus rectilineis, non ceſſat rotari niſi quatenus ab aere retardatur. Majora autem Planetarum & Cometarum corpora motus ſuos & progreſſivos & circulares in ſpatiis minus reſiſtentibus factos conſervant diutius.

Lex. II.

Mutationem motus proportionalem eſſe vi motrici impreſſæ, & fieri ſecundum lineam rectam qua vis illa imprimitur.

Si vis aliqua motum quemvis generet, dupla duplum, tripla triplum generabit, ſive ſimul & ſemel, ſive gradatim & ſucceſſive impreſſa fuerit. Et hic motus quoniam in eandem ſemper plagam cum vi generatrice determinatur, ſi corpus antea movebatur, motui ejus vel conſpiranti additur, vel contrario ſubducitur, vel obliquo oblique adjicitur, & cum eo ſecundum utriuſq; determinationem componitur.

Lex. III.

- Definition:

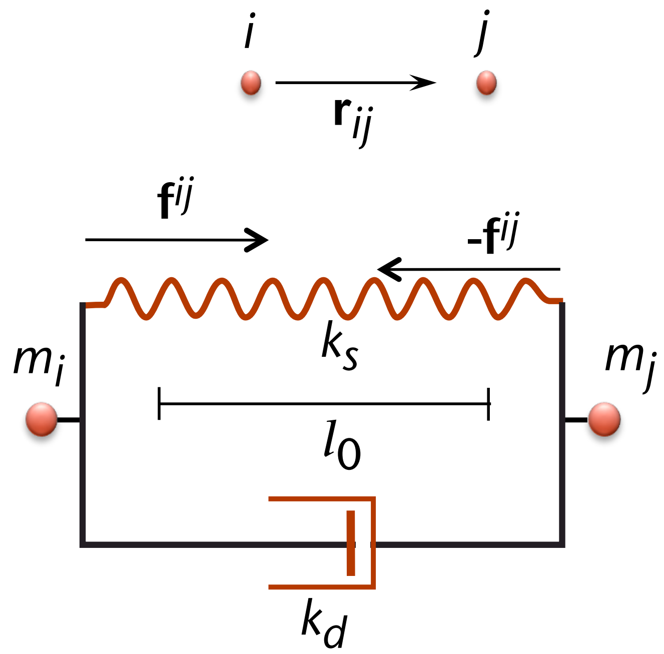
  A spring-mass-system is a system, consisting of:

  1. A set of point masses m_i with positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$, i = 1...N ;
  2. A set of springs $s_{ij} = (i, j, k_s, k_d)$, where s_{ij} connects masses *i* und *j*, with rest length l_0 , spring constant k_s (= stiffness) and the damping coeffizient k_d

- Advantages:
  - Very easy to program
  - Ideally suited to study different kinds of solving methods
  - *Ubiquitous in games* (cloths, capes, sometimes also for deformable objects)
- Disadvantages:
  - Some parameters (in particular the spring constants) are not obvious, i.e., difficult to derive
  - No volumetric effects (e.g., preservation of volume)

A Single Spring (plus Damper)

- Given: masses m_i and m_j with positions $\mathbf{x}_i$, $\mathbf{x}_j$
- Let $\mathbf{r}_{ij} = \dfrac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$
- The force between particles *i* and *j* :



1. Force extended by spring:

$$\mathbf{f}_s^{ij} = k_s \mathbf{r}_{ij} \left(\|\mathbf{x}_j - \mathbf{x}_i\| - l_0 \right)$$

acts on mass m_i in direction of m_j

2. Force extended by damper : $\mathbf{f}_d^{ij} = k_d \left((\mathbf{v}_j - \mathbf{v}_i) \cdot \mathbf{r}_{ij} \right) \mathbf{r}_{ij}$
3. Sum of forces : $\mathbf{f}^{ij} = \mathbf{f}_s^{ij} + \mathbf{f}_d^{ij}$
4. Force on m_j : $\mathbf{f}^{ji} = -\mathbf{f}^{ij}$

- Notice: (4) → the momentum is preserved
  - Momentum = force × mass = $\mathbf{F} \cdot m$
- Note on terminology:
  - German "Kraftstoß" = English "Impulse" = force × time
  - German "Impuls" = English "momentum" = force × mass
- Alternative Federkraft: $\mathbf{f}_s^{ij} = k_s \mathbf{r}_{ij} \frac{\|\mathbf{x}_j - \mathbf{x}_i\| - l_0}{l_0}$
- A spring-damper element in reality:

Simulation of a Single Spring

- From Newton's law, we have: $\ddot{\mathbf{x}} = \frac{1}{m}\mathbf{f}$
- Convert differential equation (DE) of order 2 into DE of order 1:

$$\begin{aligned} \dot{\mathbf{v}}(t) &= \frac{1}{m}\mathbf{f}(t) \\ \dot{\mathbf{x}}(t) &= \mathbf{v}(t) \end{aligned}$$

- Initial values (boundary values): $\mathbf{v}(t_0) = \mathbf{v}_0$, $\mathbf{x}(t_0) = \mathbf{x}_0$
- "Simulation" = "Integration of DE's over time"
- By Taylor expansion we get:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t\, \dot{\mathbf{x}}(t) + O\left(\Delta t^2\right)$$

- Analogeously: $\mathbf{v}\,(t + \Delta t) = \mathbf{v}\,(t) + \Delta t\, \dot{\mathbf{v}}\,(t)$

→ This integration scheme is called **explicit Euler integration**

The Algorithm

```
forall particles i :
   initialize x_i, v_i, m_i

Loop forever:
   forall particles i :
```

$$\mathbf{f}_i \leftarrow \mathbf{f}^g + \mathbf{f}_i^{coll} + \sum_{j,\,(i,j)\in S} \mathbf{f}(\mathbf{x}_i, \mathbf{v}_i, \mathbf{x}_j, \mathbf{v}_j)$$

```
   forall particles i :
```

$$\mathbf{v}_i \mathrel{+}= \Delta t \cdot \frac{\mathbf{f}_i}{m_i}$$

$$\mathbf{x}_i \mathrel{+}= \Delta t \cdot \mathbf{v}_i$$

```
   render system every n-th time
```

$\mathbf{f}^g$ = gravitational force

$\mathbf{f}^{coll}$ = penalty force exerted by collision (e.g., with obstacles)

- Advantages:
  - Can be implemented very easily
  - Fast execution per time step
- Disadvantages:
  - Stable only for very small time steps
    - Typically $\Delta t \approx 10^{-4} \ldots 10^{-3}$ sec!
  - With large time steps, additional energy is generated "out of thin air", until the system explodes ☺
  - Example: overshooting when simulating a single spring
  - Errors accumulate quickly

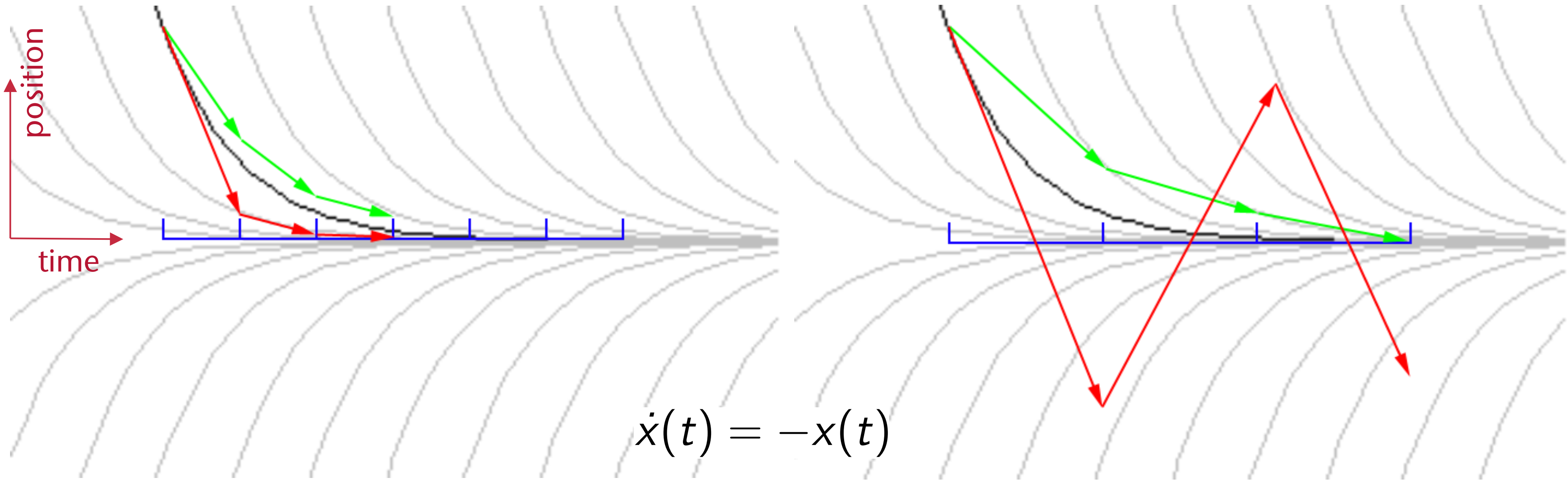
Example for the Instability of Euler Integration

- Consider the diferential equation
$$\dot{x}(t) = -kx(t)$$
- The exact solution:
$$x(t) = x_0\, e^{-kt}$$
- Euler integration does this:
$$x^{t+1} = x^t + \Delta t(-kx^t)$$
- Case $\Delta t > \frac{1}{k}$:
$$x^{t+1} = x^t \underbrace{(1 - k\Delta t)}_{<0}$$

$\Rightarrow$ x^t oscillates about 0, but approaches 0 (hopefully)

- Case $\Delta t > \frac{2}{k}$: $\Rightarrow$ $x^t \rightarrow \infty$!

- Visualization:



$$\dot{x}(t) = -x(t)$$

- Terminology: if *k* is large → the DE is called *"stiff "*
  - The stiffer the DE, the smaller Δt has to be

Visualization of Error Accumulation

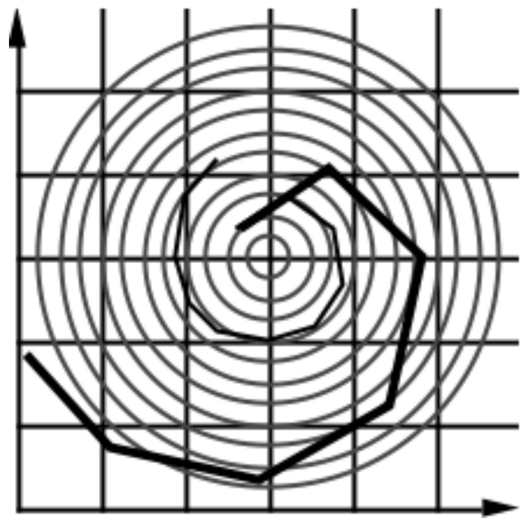
- Consider this DE:

$$\dot{\mathbf{x}}(t) = \begin{pmatrix} -x_2 \\ x_1 \end{pmatrix}$$

- Exact solution:

$$\mathbf{x}(t) = \begin{pmatrix} r\cos(t+\phi) \\ r\sin(t+\phi) \end{pmatrix}$$

- The solution by Euler integration moves in spirals outward, no matter how small Δt!
- Conclusion: Euler integration accumulates errors, no matter how small Δt!

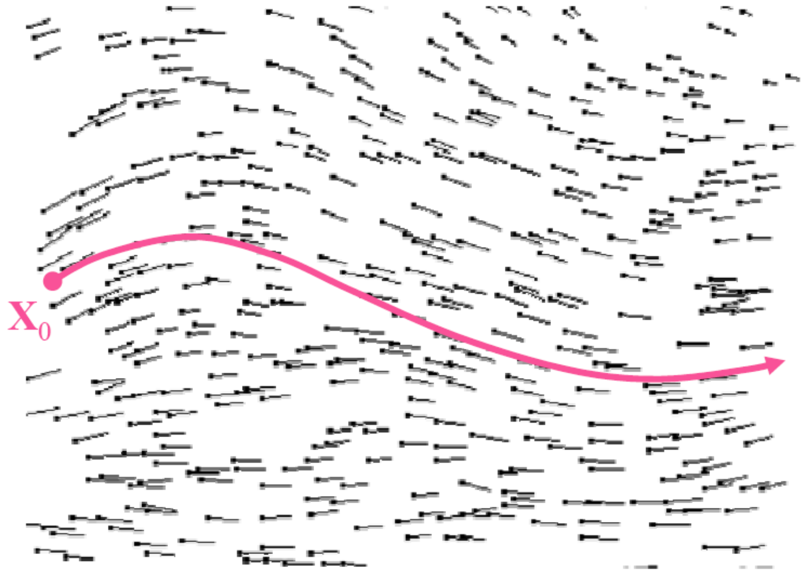


Visualization of Differential Equations

- The general form of a DE:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\,\mathbf{x}(t), t\,)$$

- Visualization of **f** as a vector field:



  - Notice: this vector field can vary over time!
- Solution of a boundary value problem = path through this field

- Runge-Kutta of order 2:
  - Idea: approximate **f**(**x**(*t*), *t*) by a quadratic function that is defined at positions **x**(*t*), **x**(*t*+ ½Δ*t*) and **v**(*t*)
  - The integrator (w/o proof):

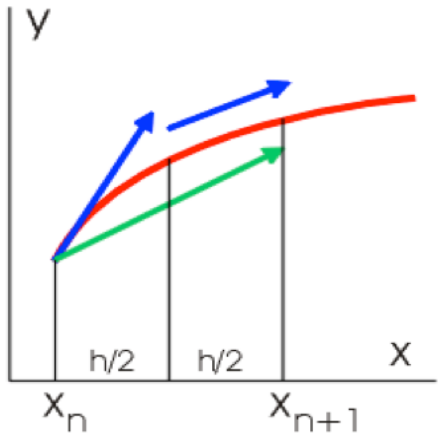
$$\mathbf{a}_1 = \mathbf{v}^t \qquad \mathbf{a}_2 = \frac{1}{m}\mathbf{f}(\mathbf{x}^t, \mathbf{v}^t)$$

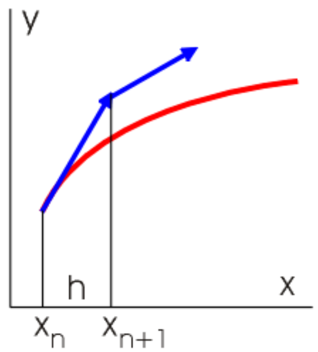
$$\mathbf{b}_1 = \mathbf{v}^t + \frac{1}{2}\Delta t \mathbf{a}_2 \qquad \mathbf{b}_2 = \frac{1}{m}\mathbf{f}\left(\mathbf{x}^t + \frac{1}{2}\Delta t \mathbf{a}_1, \mathbf{v}^t + \frac{1}{2}\Delta t \mathbf{a}_2\right)$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta t \mathbf{b}_1 \qquad \mathbf{v}^{t+1} = \mathbf{v}^t + \Delta t \mathbf{b}_2$$

- Runge-Kutta of order 4:
  - The standard integrator among the explicit integration schemata
  - Needs 4 function evaluations (i.e., force computations) per time step
  - Order of convergence is: $e(\Delta t) = O\left(\Delta t^4\right)$

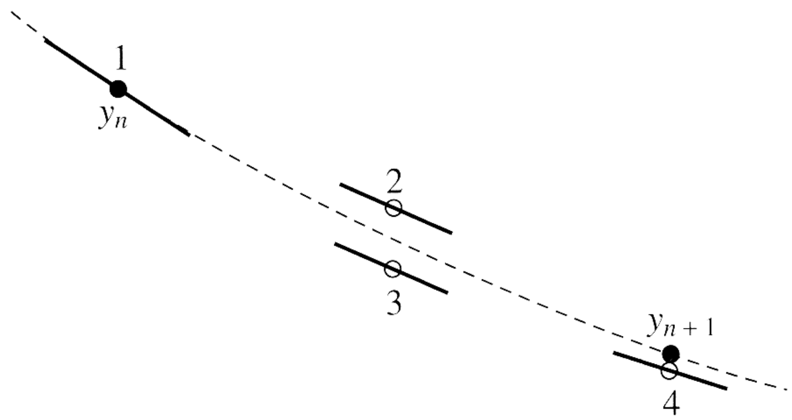
- Runge-Kutta of order 2:





Euler

- Runge-Kutta of order 4:



Verlet Integration

- A general, alternative method to increase the order of convergence: utilizes values from history
- Verlet: utilize $\mathbf{x}(t - \Delta t)$
- Derivation:
  - Develop The taylor series in both time directions:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \dot{\mathbf{x}}(t) + \frac{1}{2}\Delta t^2 \ddot{\mathbf{x}}(t) + \frac{1}{6}\Delta t^3 \dddot{\mathbf{x}}(t) + O\left(\Delta t^4\right)$$

$$\mathbf{x}(t - \Delta t) = \mathbf{x}(t) - \Delta t \dot{\mathbf{x}}(t) + \frac{1}{2}\Delta t^2 \ddot{\mathbf{x}}(t) - \frac{1}{6}\Delta t^3 \dddot{\mathbf{x}}(t) + O\left(\Delta t^4\right)$$

- Add both:

$$\mathbf{x}(t+\Delta t)+\mathbf{x}(t-\Delta t) = 2\mathbf{x}(t)+\Delta t^2\,\ddot{\mathbf{x}}(t)+O(\Delta t^4)$$

$$\mathbf{x}(t+\Delta t) = 2\mathbf{x}(t)-\mathbf{x}(t-\Delta t)+\Delta t^2\,\ddot{\mathbf{x}}(t)+O(\Delta t^4)$$

- Initialization:

$$\mathbf{x}(\Delta t) = \mathbf{x}(0)+\Delta t\mathbf{v}(0)+\frac{1}{2}\Delta t^2\Big(\frac{1}{m}\mathbf{f}(\mathbf{x}(0),\mathbf{v}(0))\Big)$$

- Remark: the velocity does not occur an more (explicitely)

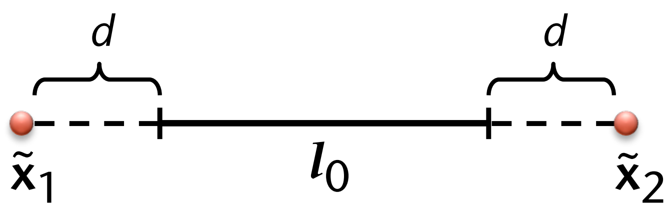
Constraints

- Big advantage of Verlet over Euler & Runge-Kutta: it is very easy to handle constraints
- Definition: Constraint = some condition on position of one or more mass points
- Examples:
  1. A point must not penetrate an obstacle
  2. The distance between two points must be constant, or distance must be $\leq$ some specific distance

- Examples:
  - Consider the constraint:

$$\|\mathbf{x}_1 - \mathbf{x}_2\| \stackrel{!}{=} l_0$$

1. Perform one Verlet integration step → $\tilde{\mathbf{x}}^{t+1}$
2. Enforce the constraint:



$$\mathbf{x}_1^{t+1} = \tilde{\mathbf{x}}_1^{t+1} + \frac{1}{2}\mathbf{r}_{12} \cdot \left(\|\tilde{\mathbf{x}}_2^{t+1} - \tilde{\mathbf{x}}_1^{t+1}\| - l_0 \right)$$

$$\mathbf{x}_2^{t+1} = \tilde{\mathbf{x}}_2^{t+1} - \underbrace{\frac{1}{2}\mathbf{r}_{12} \cdot \left(\|\tilde{\mathbf{x}}_2^{t+1} - \tilde{\mathbf{x}}_1^{t+1}\| - l_0 \right)}_{\mathbf{d}}$$

- Problem: if several constraints are to constrain the same mass point, we need to employ constraint algorithms