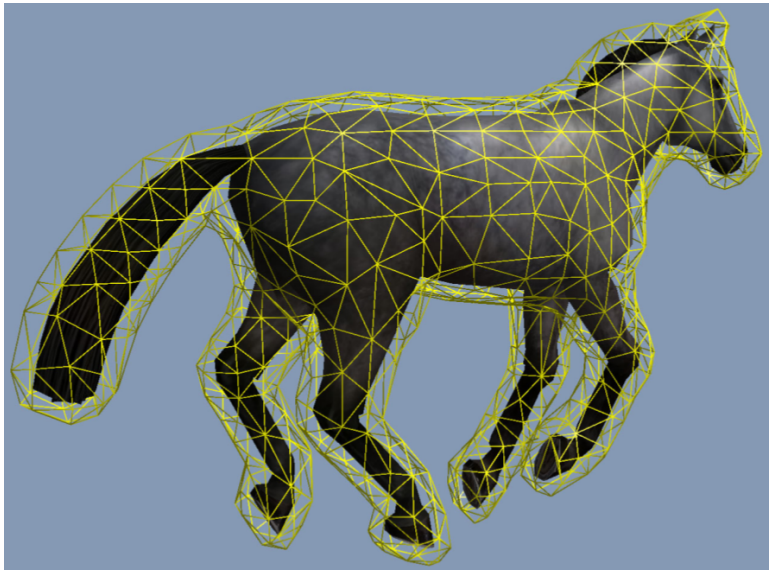
Bremen

CG VR

Virtuelle Realität
Feder-Masse-Systeme



G. Zachmann

University of Bremen, Germany

cgvr.cs.uni-bremen.de

Die Newton'schen Gesetze

1. Gesetz (Trägheitsgesetz):

> Ein kräftefreier Körper bewegt sich geradlinig mit konstanter Geschwindigkeit weiter.

- Ein Körper in Ruhe ist also nur ein Spezialfall hiervor.

2. Gesetz (Aktionsprinzip):

> Wenn eine Kraft **F** auf einen Körper mit Masse m wirkt, beschleunigt sie diesen gemäß
>
> $$\mathbf{F} = m \cdot \mathbf{a}$$

- Mit anderen Worten: Kraft und Beschleunigung sind proportional zueinander; die Proportionalitätskonstante ist m. Insbesondere haben beide dieselbe Richtung.

3. Gesetz (Reaktionsprinzip):

Wenn die Kraft **F**, die auf einen Körper wirkt, ihren Ursprung in einem anderen Körper hat, so wirkt auf diesen die entgegengesetzte Kraft $-\mathbf{F}$.

- In der Schule lernt man : "Kraft = Gegenkraft"

4. Gesetz (Superpositionsprinzip):

Wirken auf einen Punkt (oder einen starren Körper) mehrere Kräfte $\mathbf{F}_1, \ldots, \mathbf{F}_n$, so addieren sich diese vektoriell zu einer resultierenden Kraft **F** auf:

$$\mathbf{F} = \mathbf{F}_1 + \ldots + \mathbf{F}_n .$$

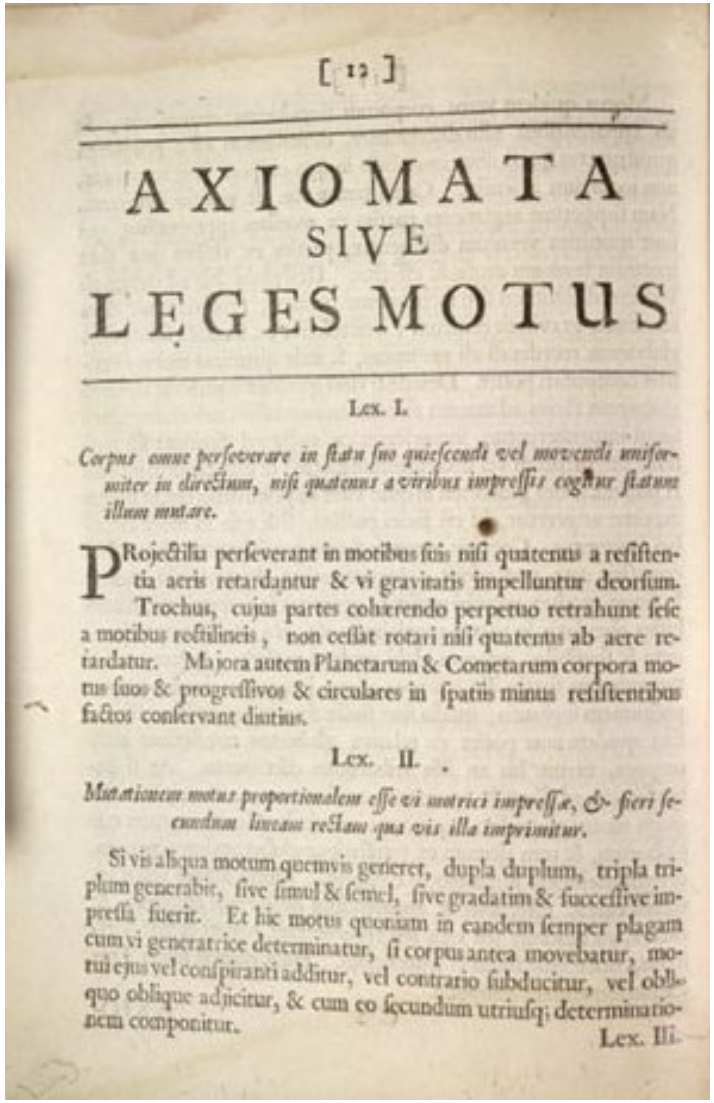
Historischer Exkurs

- Newton's Gesetze in der Originalschrift

  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.*



[13]

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.

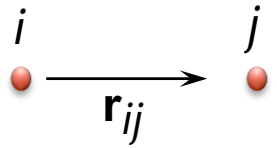
Si vis aliqua motum quemvis generet, dupla duplum, tripla triplum generabit, sive simul & semel, sive gradatim & successive impressa fuerit. Et hic motus quoniam in eandem semper plagam cum vi generatrice determinatur, si corpus antea movebatur, motui ejus vel conspiranti additur, vel contrario subducitur,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

Lex. III.

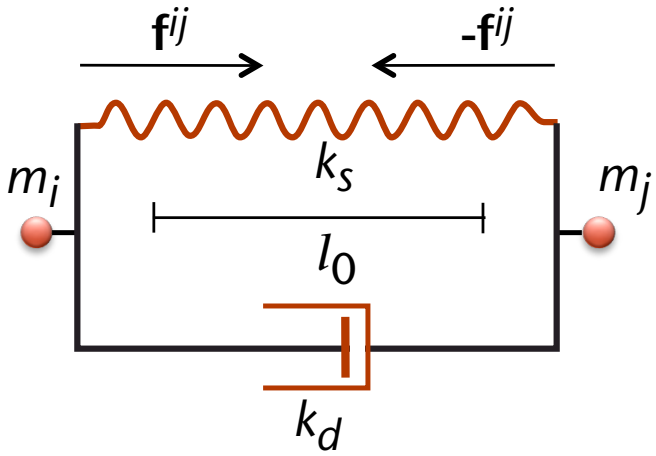
- Definition:
  Ein Feder-Masse-System ist ein System, bestehend aus:
  1. einer Menge von Punktmassen m_i mit Positionen $\mathbf{x}_i$ und Geschwindigkeit $\mathbf{v}_i$, i = 1...N ;
  2. einer Menge von Federn $s_{ij} = (i, j, k_s, k_d)$, die die Massen i und j verbindet, mit der Ruhelänge l_0, Federkonstante (= Steifigkeit) k_s und den Dämpfungskoeffizienten k_d
- Vorteile:
  - Sehr einfach zu programmieren
  - Ideal zum Studium verschiedener numerischer Lösungsverfahren
  - *Ubiquitous in games* (cloths, capes, teilweise auch deformierbare Obje, ...)
- Nachteile:
  - Einige Parameter (Federkonstanten z.B.) sind nicht offensichtlich
  - Keine volumetrischen Effekte (z.B. Volumenerhaltung)

Eine einzelne Feder (mit Dämpfer)

- Gegeben: Massen m_i und m_j mit Positionen $\mathbf{x}_i$, $\mathbf{x}_j$
- Definiere $$\mathbf{r}_{ij} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$$



- Kräfte zwischen den beiden Massen :



1. Von der Feder:

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

   wirkt auf Masse m_i in Richtung m_j

2. Durch den Dämpfer: $\mathbf{f}_d^{ij} = k_d((\mathbf{v}_j - \mathbf{v}_i)\cdot\mathbf{r}_{ij})\mathbf{r}_{ij}$
3. Zusammen : $\mathbf{f}^{ij} = \mathbf{f}_s^{ij} + \mathbf{f}_d^{ij}$
4. Kraft auf m_j : $\mathbf{f}^{ji} = -\mathbf{f}^{ij}$

- Aus (4) → der Impuls bleibt erhalten!

- Alternative Federkraft:

$$\mathbf{f}_s^{ij} = k_s \mathbf{r}_{ij} \frac{\|\mathbf{x}_j - \mathbf{x}_i\| - l_0}{l_0}$$

- Ein Feder-Dämpfer in der Realität:

Simulation einer einzelnen Feder

- Aus Newton'schen Gesetzen folgt: $\ddot{\mathbf{x}} = \frac{1}{m}\mathbf{f}$
- DGL der Ordnung 2 umwandeln in DGLs erster Ordnung:

$$\dot{\mathbf{v}}(t) = \frac{1}{m}\mathbf{f}(t)$$

$$\dot{\mathbf{x}}(t) = \mathbf{v}(t)$$

- Anfangswerte: $\mathbf{v}(t_0) = \mathbf{v}_0\ ,\quad \mathbf{x}(t_0) = \mathbf{x}_0$
- "Simulation" = "Integration von DGLs über die Zeit"
- Taylor-Expansion liefert:

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

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

→ Dieses Integrationsverfahren heißt **explizite Euler-Integration**

Der Algorithmus

forall particles *i* :
  initialize $\mathbf{x}_i$, $\mathbf{v}_i$, m_i

loop:
  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* :

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

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

  render system every *n*-th time

$\mathbf{f}^g$ = Gravitationskraft

$\mathbf{f}^{coll}$ = Rückstoßkraft aus Kollision (z.B. mit Hindernis)

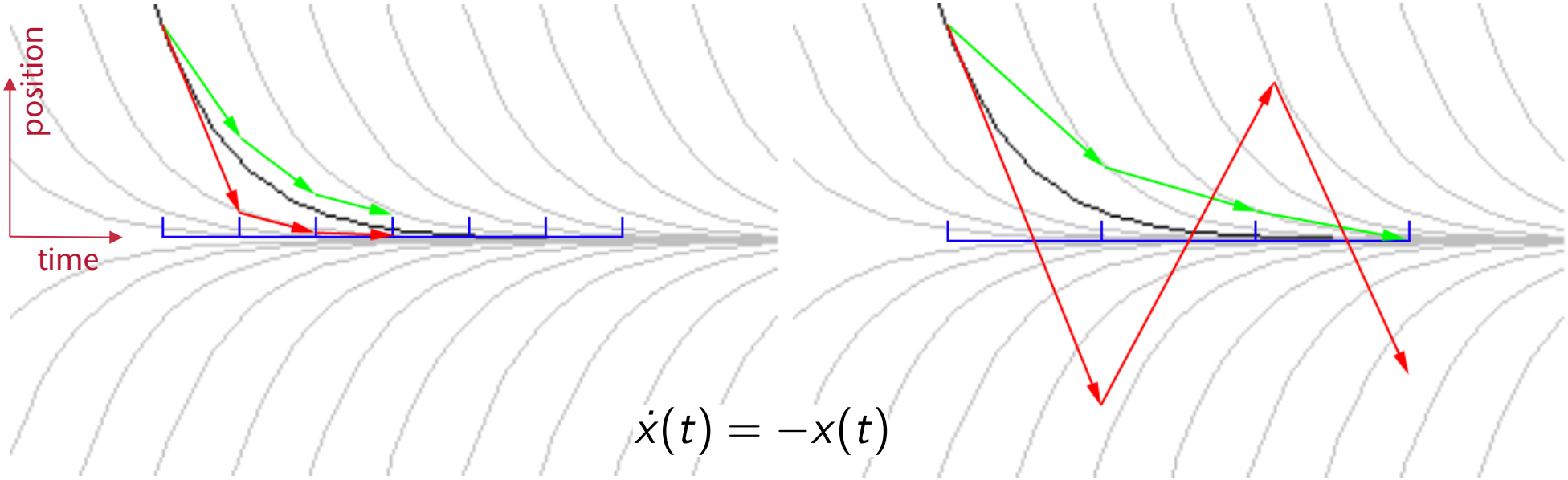
- Weitere Integrations-Schemata:
  - Midpoint
  - ...
- Vorteil: einfach zu implementieren, schnelle Ausführung pro Zeitschritt
- Nachteil: nur für sehr kleine Schrittweiten stabil
  - Typ.weise $\Delta t \approx 10^{-4} \ldots 10^{-3}$ sec!
  - Bei großen Schrittweiten wird zusätzliche Energie im System erzeugt, und schließlich explodiert es ☺
  - Beispiel: Overshooting bei einfacher Feder
- Weiterer Nachteil: Fehler akkumulieren sich schnell (bei Euler)

Beispiel für die Instabilität des Euler-Integrationsverfahrens

- Betrachte die DGL
$$\dot{x}(t) = -kx(t)$$
- Die exakte Lösung:
$$x(t) = x_0\, e^{-kt}$$
- Das Euler-Verfahren liefert:
$$x^{t+1} = x^t + \Delta t(-kx^t)$$
- Falls $\Delta t > \frac{1}{k}$:
$$x^{t+1} = x^t \underbrace{(1 - k\Delta t)}_{<0}$$

  $\Rightarrow$ x^t oszilliert um 0, geht aber (hoffentlich) gegen 0
- Falls $\Delta t > \frac{2}{k}$ $\Rightarrow$ $x^t \rightarrow \infty$!

- Visualisierung:



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

- Bezeichnung: falls k groß → "steife DGL" ("*stiff ODE*")
  - Dann muß Δt sehr klein sein

Veranschaulichung der Fehlerakkumulation

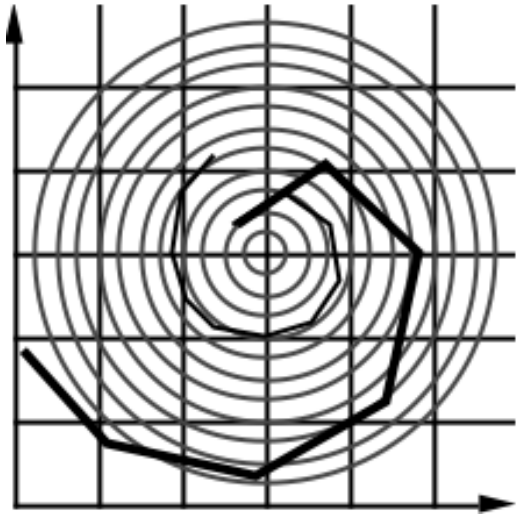
- Betrachte die DGL:

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

- Exakte Lösung:

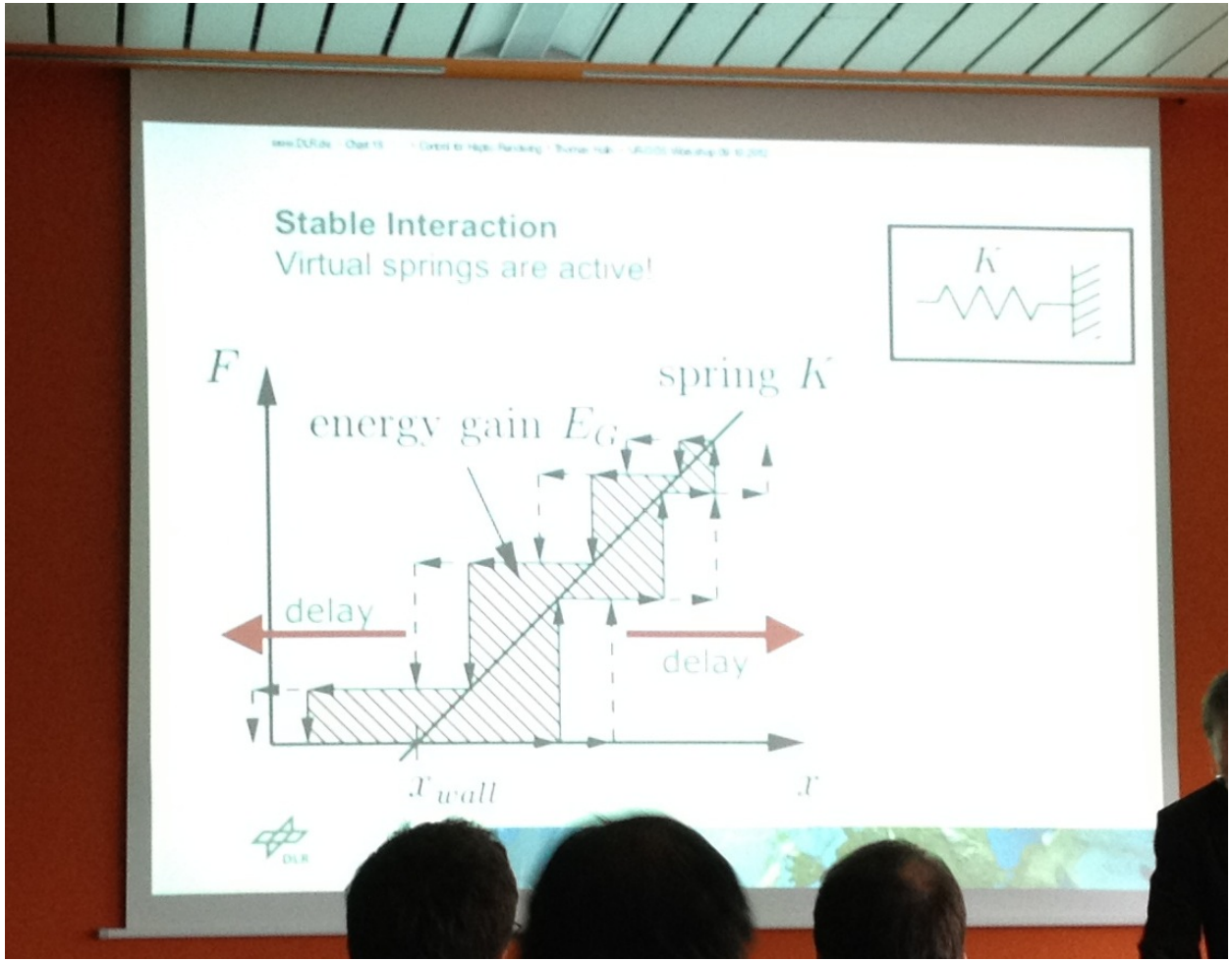
$$\mathbf{x}(t) = \begin{pmatrix} r\cos(t+\phi) \\ r\sin(t+\phi) \end{pmatrix}$$

- Euler läuft in Spiralen nach außen, egal wie klein die Schrittweite gewählt wird!
- Fazit: Euler-Integration akkumuliert Fehler, egal wie klein die Schrittweite ist!



Weiterer Grund für Instabilitäten

- Durch diskrete Simulationsschritte kann Energie im System entstehen:

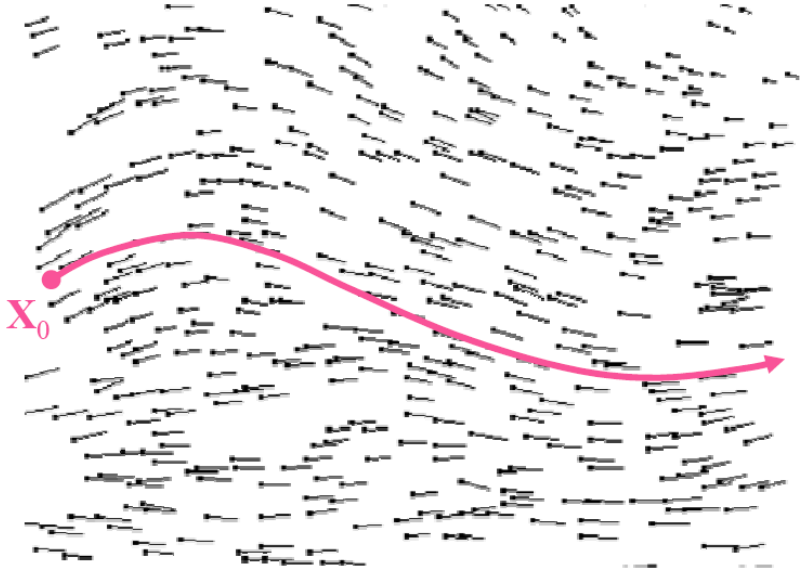


Veranschaulichung von DGLs

- Die allgemeine Form einer DGL (hier):

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

- Visualisiere **f** als Vektorfeld:



  - Achtung: dieses Vektorfeld kann sich mit *t* ändern!
- Lösung eines Anfangswertproblems = Pfad durch dieses Feld

- Runge-Kutta 2-ter Ordnung:
  - Idee: approximiere **f**(**x**(*t*), *t*) durch quadratische Funktion, die an der Stelle **x**(*t*), **x**(*t*+ ½Δ*t*) und durch **v**(*t*) definiert wird
  - Der Integrator (o. Bew.):

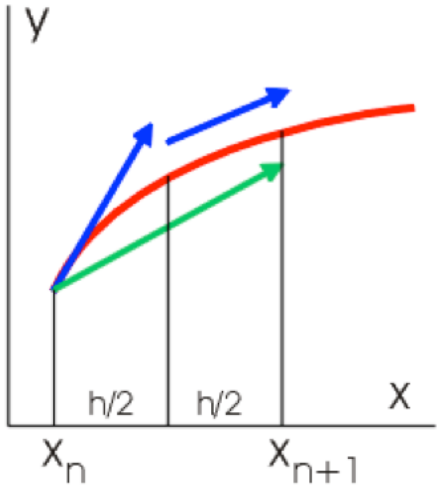
$$\mathbf{a}_1 = \mathbf{v}^t \qquad \mathbf{a}_2 = \frac{1}{m}\mathbf{f}(\mathbf{x}^t, \mathbf{v}^t)$$

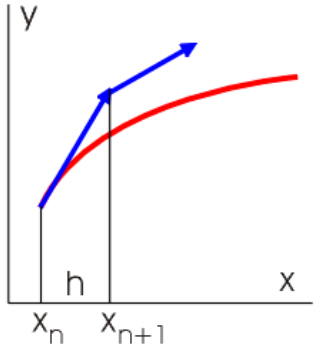
$$\mathbf{b}_1 = \mathbf{v}^t + \frac{1}{2}\Delta t\mathbf{a}_2 \qquad \mathbf{b}_2 = \frac{1}{m}\mathbf{f}\left(\mathbf{x}^t + \frac{1}{2}\Delta t\mathbf{a}_1, \mathbf{v}^t + \frac{1}{2}\Delta t\mathbf{a}_2\right)$$

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

- Runge-Kutta 4-ter Ordnung:
  - Der Standard-Integrator unter den expliziten Integrations-Schemata
  - Benötigt 4 Funktionsauswertungen (Kräfte berechnen) pro Zeitschritt
  - Konvergenzordnung: $e(\Delta t) = O(\Delta t^4)$

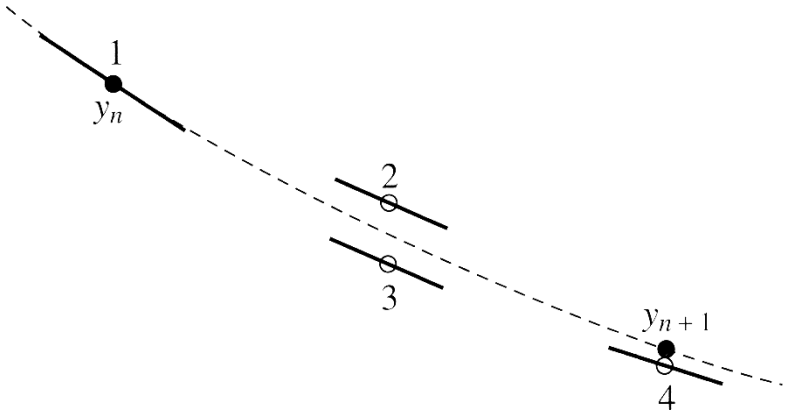
- Runge-Kutta 2-ter Ordnung:





Euler

- Runge-Kutta 4-ter Ordnung:



Verlet-Integration

- Alternative Methode zur Steigerung der Konvergenzordnung: verwende Werte aus der Vergangenheit
- Verlet: verwendet $\mathbf{x}(t - \Delta t)$
- Herleitung:
  - Taylor-Reihe in beide Zeitrichtungen entwickeln:

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

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

- Addieren:

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

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

- Initialisierung:

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

- Bemerkung: Geschwindigkeit taucht nicht mehr (explizit) auf

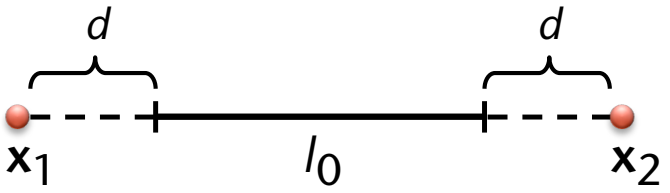
Constraints

- Großer Vorteil gegenüber Euler & Runge-Kutta:
  man kann sehr leicht Constraints behandeln
- Definition: Constraint = Einschränkung der Position eines oder mehrerer Massenpunkte
- Beispiele:
  1. Ein Massenpunkt darf nicht in ein Hindernis eindringen
  2. Der Abstand zwischen zwei Massenpunkten soll konstant sein, oder ≤ bestimmter Abstand

- Verfahren anhand eines Beispiels:
  - Constraint:

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

1. Führe einen Verlet-Integrationsschritt durch → $\tilde{\mathbf{x}}^{t+1}$
2. Enforce Constraint:



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

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

- Problem, falls mehrere Constraints denselben Massepunkt einschränken sollen
  - Verwende Constraint-Algorithmen