Embodied Vision

Physical Modeling: Time Integration, Mass-Spring System and Finite Element Method

Tsung-Wei Ke

Spring 2026

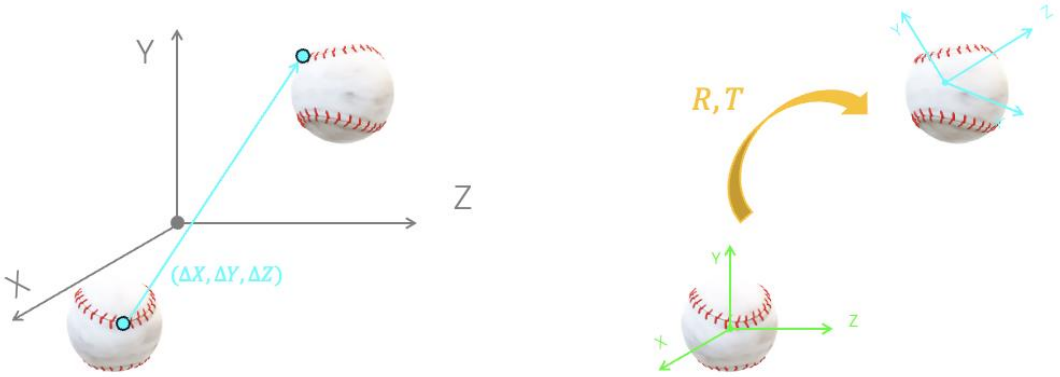
Disclaimer

- This lecture borrows contents heavily from
  - Physics-based Animation by David I.W Levin at University of Toronto
  - Physics-based Animation of Solids and Fluids by Minchen Li at CMU

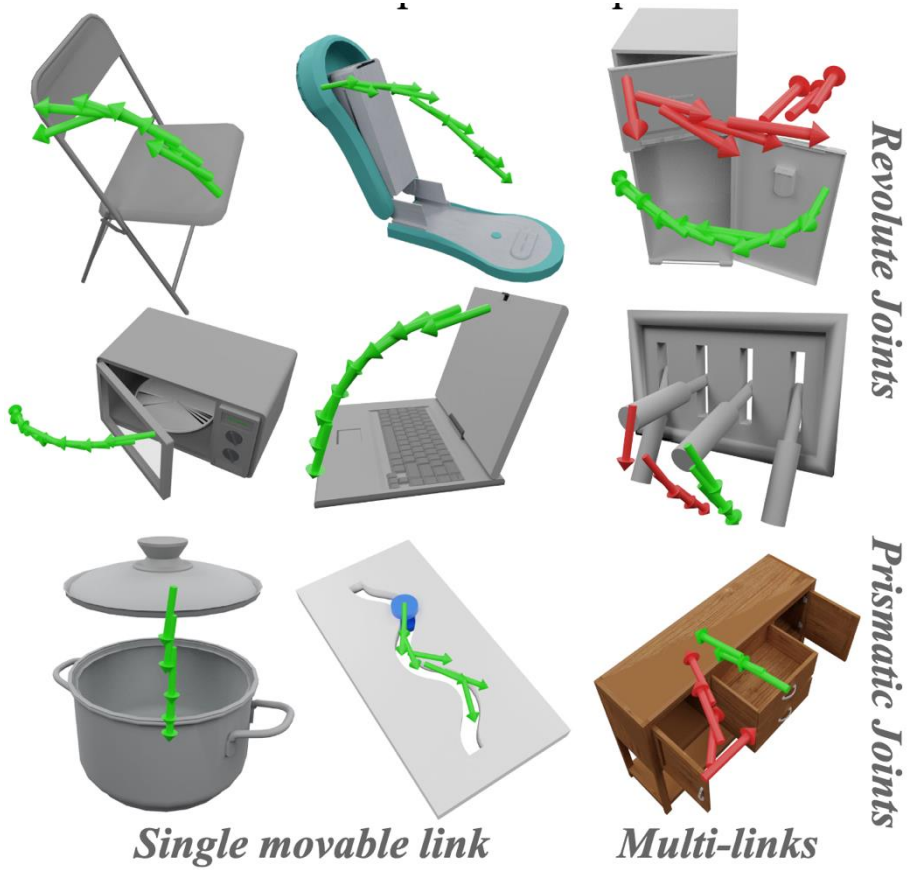
- This is an introductory lecture to physical modeling. Check the advanced courses in physic-based simulation for more details.

Recap

Rigid-body Motion



Articulated-object Motion



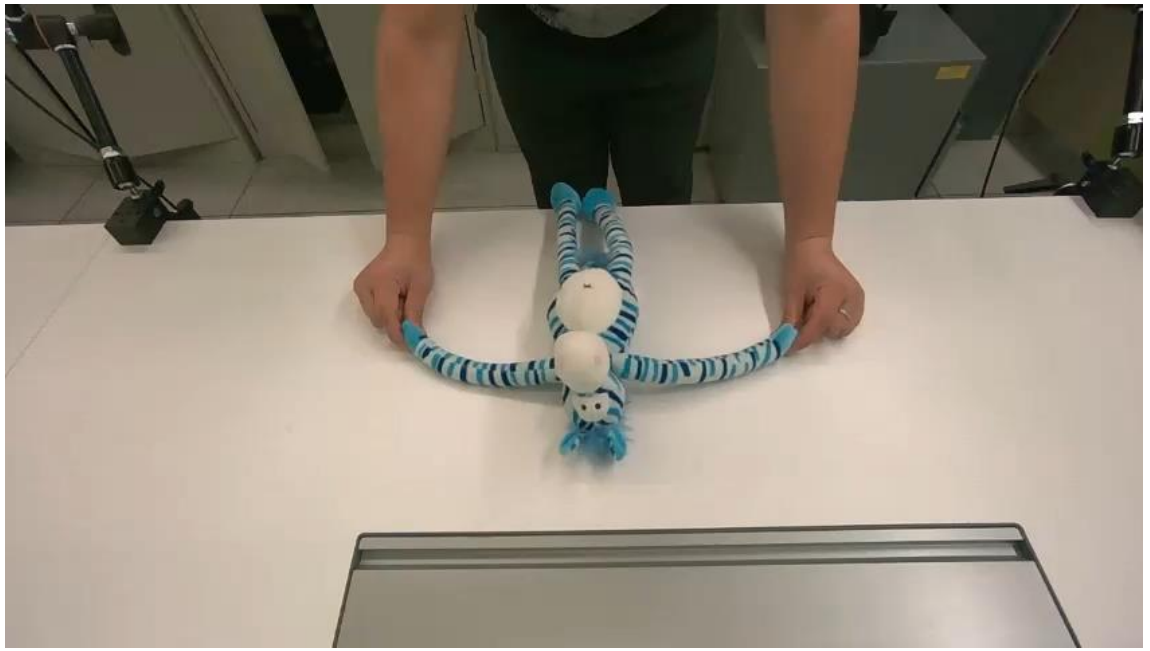
Rigid-body physic law:

$$\frac{d}{dt}\mathbf{q}(t) = \frac{d}{dt}\begin{bmatrix}\mathbf{t}(t)\\ \mathbf{R}(t)\\ \boldsymbol{\nu}(t)\\ \boldsymbol{\omega}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{v}(t)\\ \boldsymbol{\omega}(t)\times\mathbf{R}(t)\\ \frac{\mathbf{F}(t)}{M}\\ \mathbf{I}(t)^{-1}(\boldsymbol{\tau}-\boldsymbol{\omega}(t)\times\mathbf{I}(t)\boldsymbol{\omega}(t))\end{bmatrix}$$

Simulation:

$$\mathbf{q}(t) = \mathbf{q}(0) + \int_0^t \frac{d}{dt}\mathbf{q}(t)dt = \mathbf{q}(0) + \sum_{i=1}^{T}\frac{d}{dt}\mathbf{q}(t)|_{t=t_i}\Delta_t$$

However, Most Objects are not Rigid Bodies…



However, Most Objects are not Rigid Bodies…

Cloth

Rope

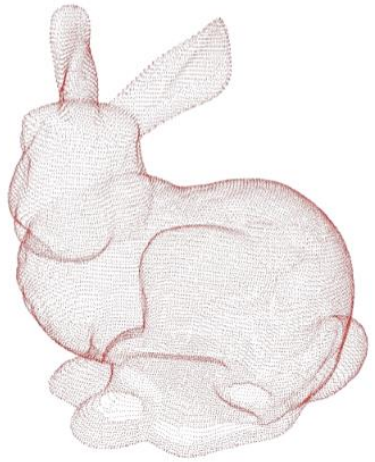
Fluid

Soft body

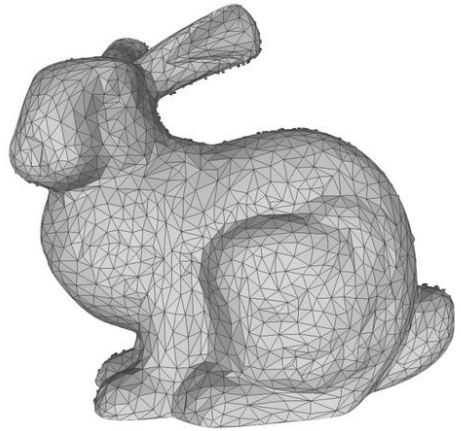
Image source: https://obi.virtualmethodstudio.com/

How to Physically Model Non-Rigid-Body Motion?

1. How to represent 3D model of non-rigid bodies?



point

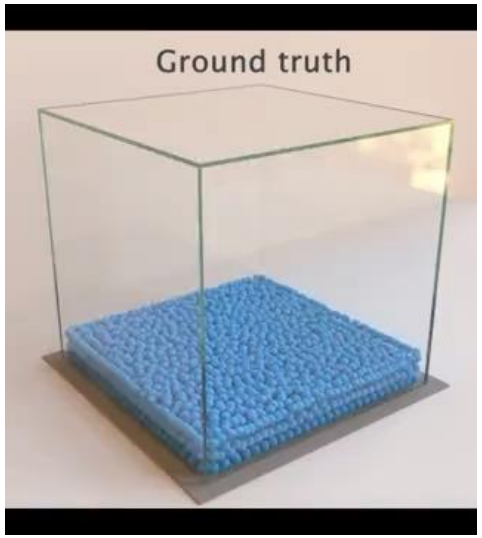


mesh

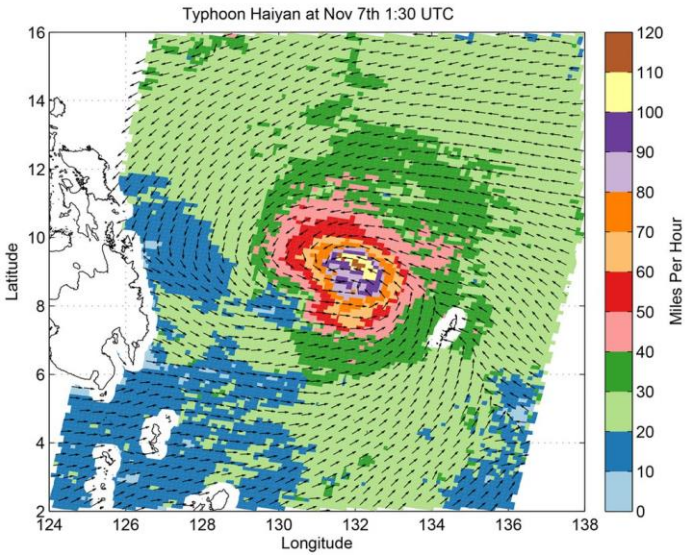
Image source: https://obi.virtualmethodstudio.com/

How to Physically Model Non-Rigid-Body Motion?

1. How to represent 3D model of non-rigid bodies?
2. How to represent non-rigid-body motion?



Particle motion

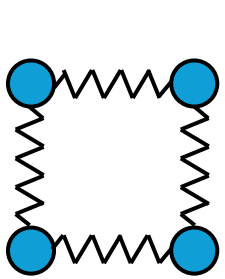


Motion field

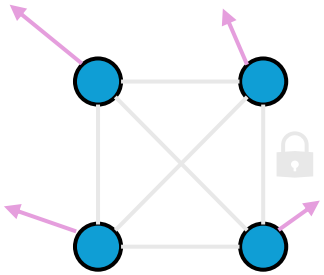
Image source: https://obi.virtualmethodstudio.com/

How to Physically Model Non-Rigid-Body Motion?

1. How to represent 3D model of non-rigid bodies?
2. How to represent non-rigid-body motion?
3. How to physically model non-rigid-body motion?
   Note that point motions are correlated



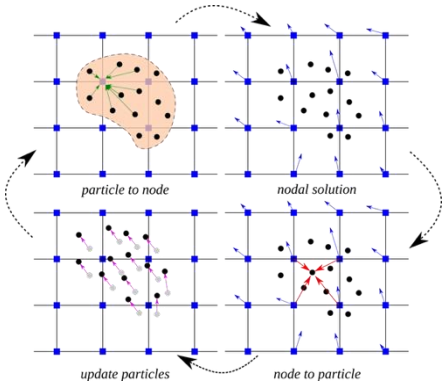
Mass-spring system



Position Based Dynamics

$$\min_{x} E(x, x^t, v^t) \quad \text{s.t.} \quad g(x) \geq 0.$$

Incremental Potential Solver



Material Point Method

Image source: https://andrewdcampbell.github.io/clothsim/

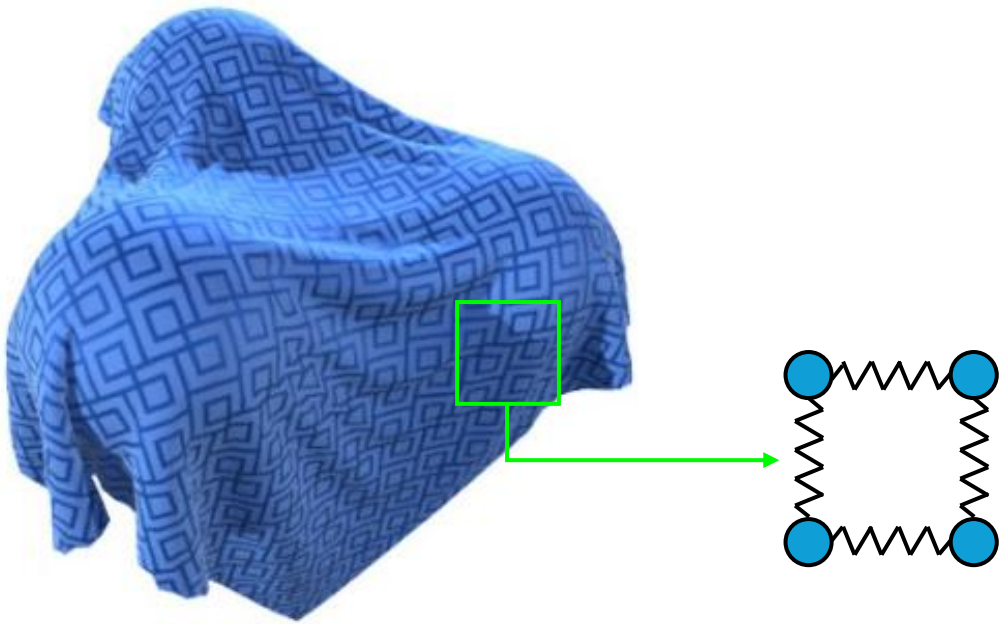
Content

- Non-Rigid Modeling
  - ➢Mass-spring system
  - ➢Position-based Dynamics
  - ➢Tetrahedral and Deformation

Content

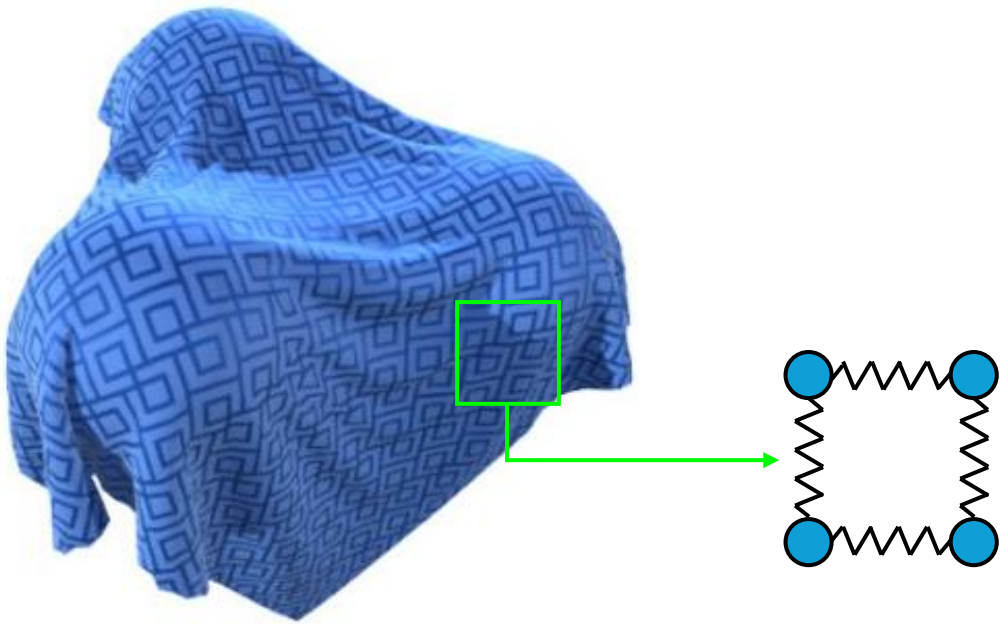
- Non-Rigid Modeling
  - ➢Mass-spring system
  - ➢Position-based Dynamics
  - ➢Tetrahedral and Deformation

Mass-Spring System



- **Idea**: Represent a deformable object as a set of mass nodes $\mathcal{V}$ connected by spring-based edges $\mathcal{E}$, forming a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- The connectivity pattern is pre-defined
- The dynamic formulation of a mass-spring system:
  - Calculate the total force on each node
  - Update the velocity and position of each node

Mass-Spring System



- The dynamic formulation of a mass-spring system:
  - The total force on node i:

$$F_i = \sum_{i,j \in \mathcal{E}} F_{i,j}^{spring} + F_{i,j}^{dashpot} + F_i^{ext}$$

  - Spring force describes the bounding force between nodes:

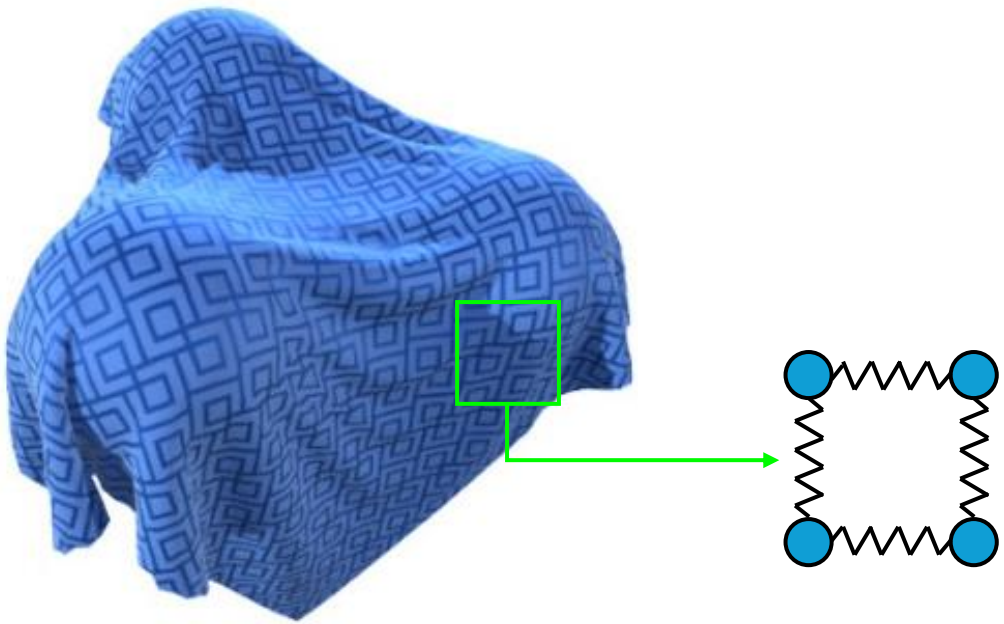
$$F_{i,j}^{spring} = k_{i,j}\left(\|x_j - x_i\| - l_{i,j}\right)\frac{x_j - x_i}{\|x_j - x_i\|}$$

  - The dashpot dissipates energy, preventing oscillations:

$$F_{i,j}^{dashpot} = -\gamma(\|v_i - v_i\|)$$

  - External forces accounts for collision, gravity and user interactions

Mass-Spring System



- The dynamic formulation of a mass-spring system:
  - The total force on node i:

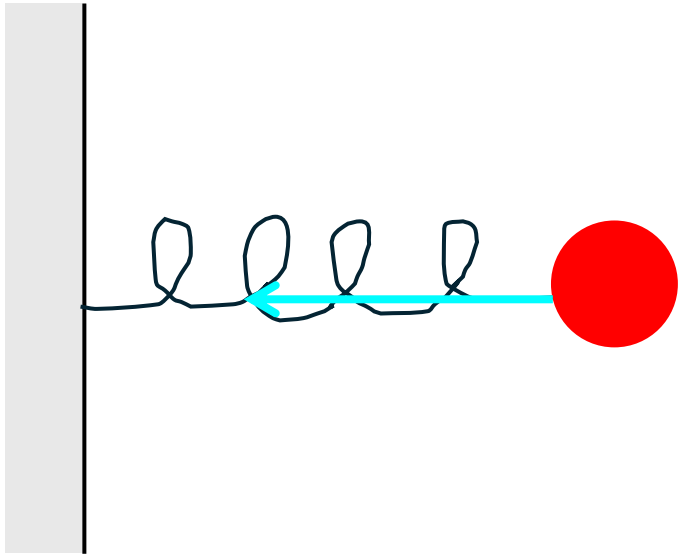
$$F_i = \sum_{i,j \in \mathcal{E}} F_{i,j}^{spring} + F_{i,j}^{dashpot} + F_i^{ext}$$

  - The updated velocity and position of node i:

$$v_i^{t+1} = \delta\left(v_i^t + \Delta t \frac{F_i}{m_i}\right)$$
$$x_i^{t+1} = x_i^t + \Delta t v_i^{t+1}$$

How do we derive this update rule?
Is this update rule optimal?

Let's Consider a Simple Case



- The total force on node i:

$$F_i = \sum_{i,j \in \mathcal{E}} F_{i,j}^{spring} + F_{i,j}^{dashpot} + F_i^{ext}$$

- Another end is attached to the wall: $x_j = 0$
- The rest length of the spring is zero: $l_{i,j} = 0$
- Spring force:

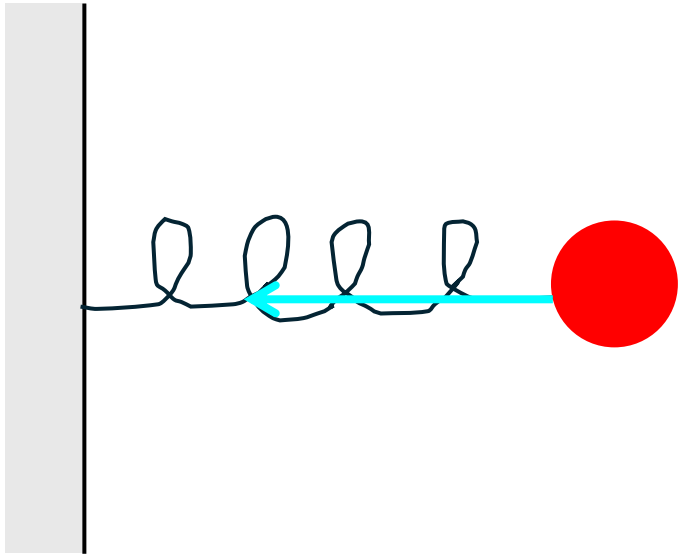
$$F_i^{spring} = -kx_i(t)$$

- Newton's second law:

$$F_i = m\ddot{x}_i(t) \Rightarrow \underbrace{-kx_i(t) = m\ddot{x}_i(t)}_{2^{nd} \text{ order ODE}}$$

Slide adapted from D. Levin

Idea: Simplify One 2nd Order ODE to Two 1st Order ODE



- 2nd order ODE:

$$-kx_i(t) = m\ddot{x}_i(t)$$

- We know that:

$$\ddot{x} = \dot{v}$$

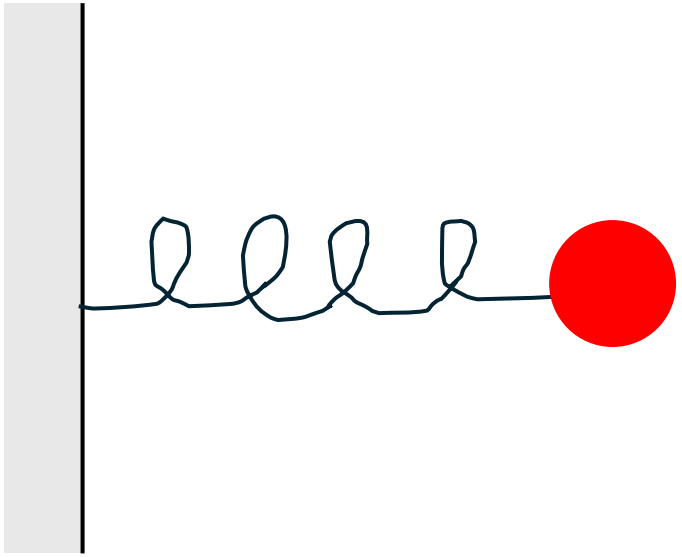
$$\dot{x} = v$$

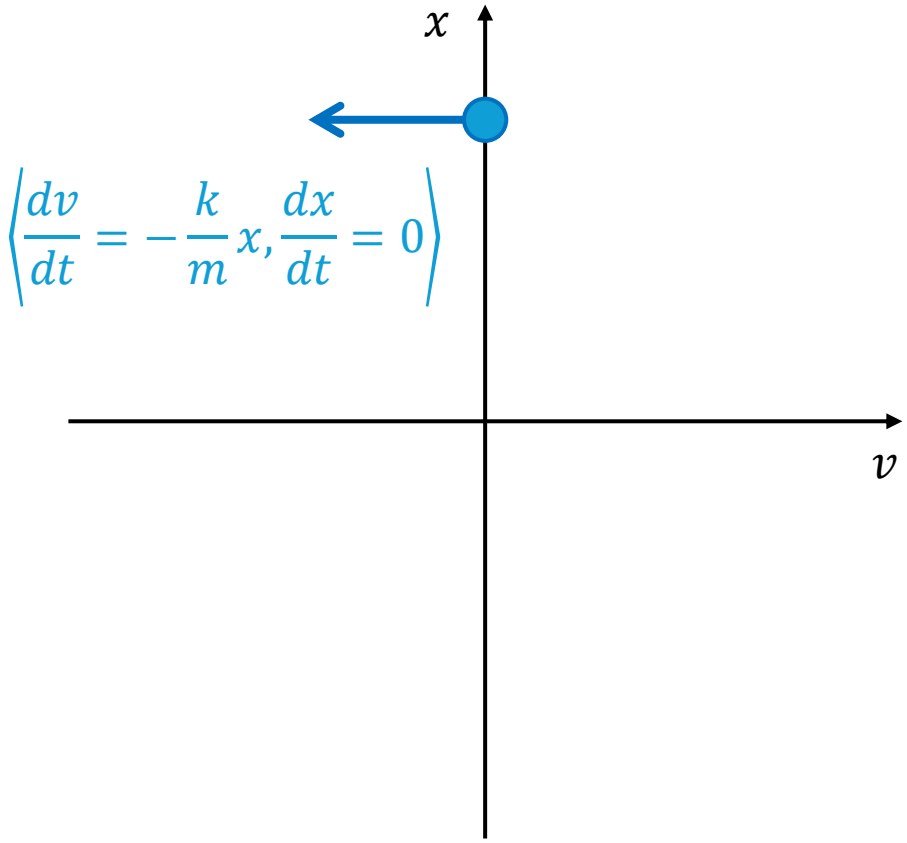
- We convert a 2nd order ODE to two 1st order ODEs:

$$\underbrace{\begin{pmatrix} m & 0 \\ 0 & 1 \end{pmatrix}}_{A} \underbrace{\frac{d}{dt}\begin{pmatrix} v \\ x \end{pmatrix}}_{\dot{y}} = \underbrace{\begin{pmatrix} 0 & -k \\ 1 & 0 \end{pmatrix} \overbrace{\begin{pmatrix} v \\ x \end{pmatrix}}^{y}}_{f(y)}$$

Slide adapted from D. Levin

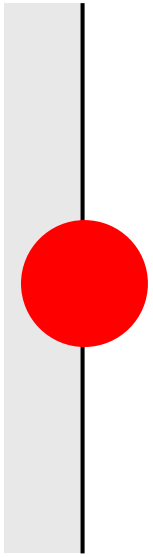
Idea: Simplify One 2nd Order ODE to Two 1st Order ODE

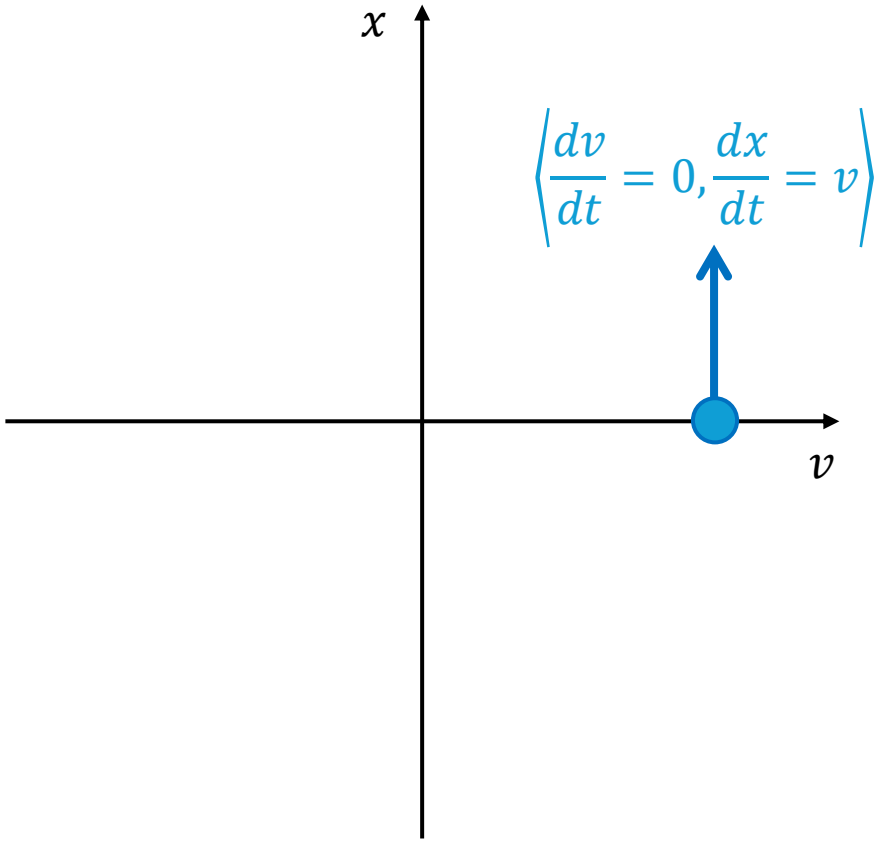




$$\begin{pmatrix} m & 0 \\ 0 & 1 \end{pmatrix} \frac{d}{dt} \begin{pmatrix} v \\ x \end{pmatrix} = \begin{pmatrix} 0 & -k \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ x \end{pmatrix}$$

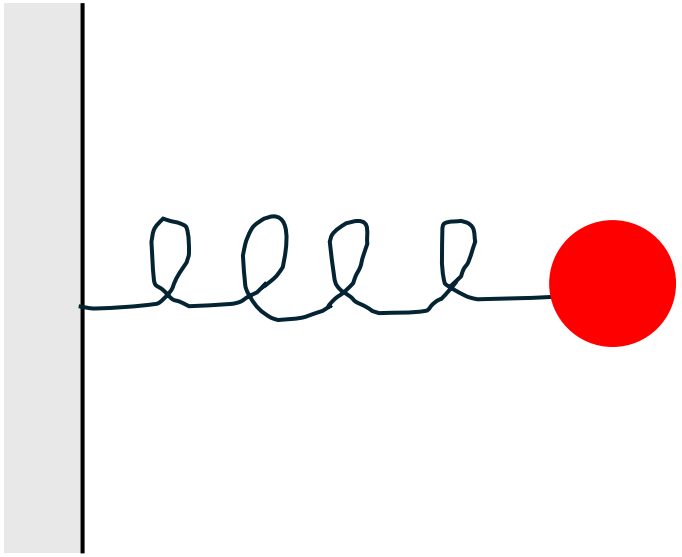
Idea: Simplify One 2nd Order ODE to Two 1st Order ODE

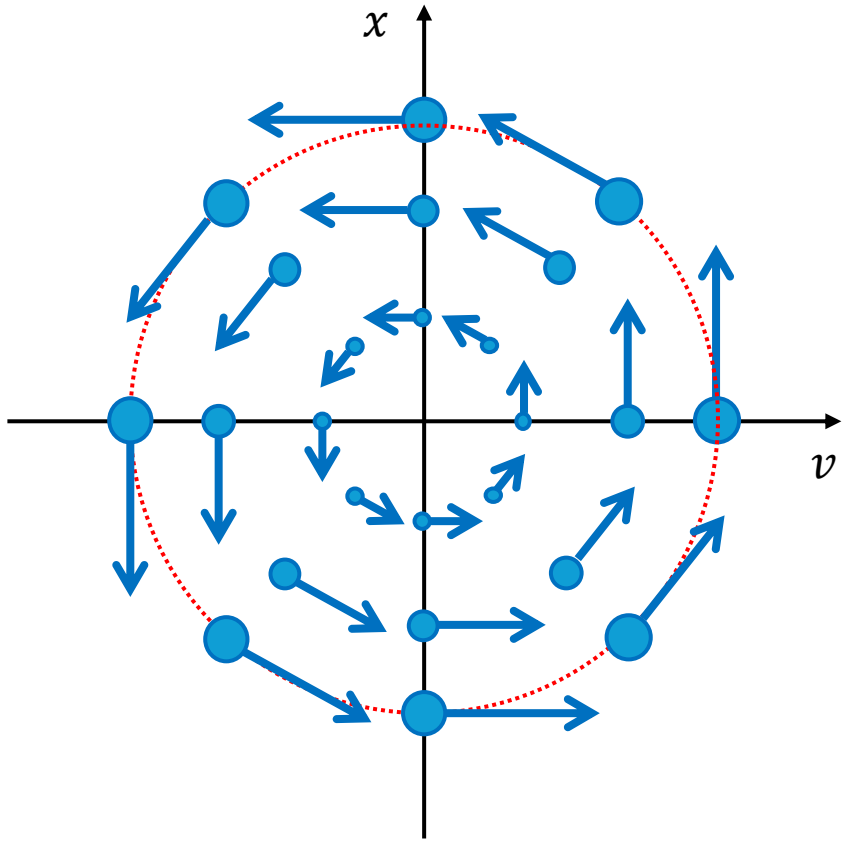




$$\begin{pmatrix} m & 0 \\ 0 & 1 \end{pmatrix} \frac{d}{dt} \begin{pmatrix} v \\ x \end{pmatrix} = \begin{pmatrix} 0 & -k \\ 1 & 0 \end{pmatrix} \begin{pmatrix} v \\ 0 \end{pmatrix}$$

Idea: Simplify One 2nd Order ODE to Two 1st Order ODE





$$\begin{pmatrix} m & 0 \\ 0 & 1 \end{pmatrix} \frac{d}{dt} \begin{pmatrix} v \\ x \end{pmatrix} = \begin{pmatrix} 0 & -k \\ 1 & 0 \end{pmatrix} \begin{pmatrix} v \\ x \end{pmatrix}$$

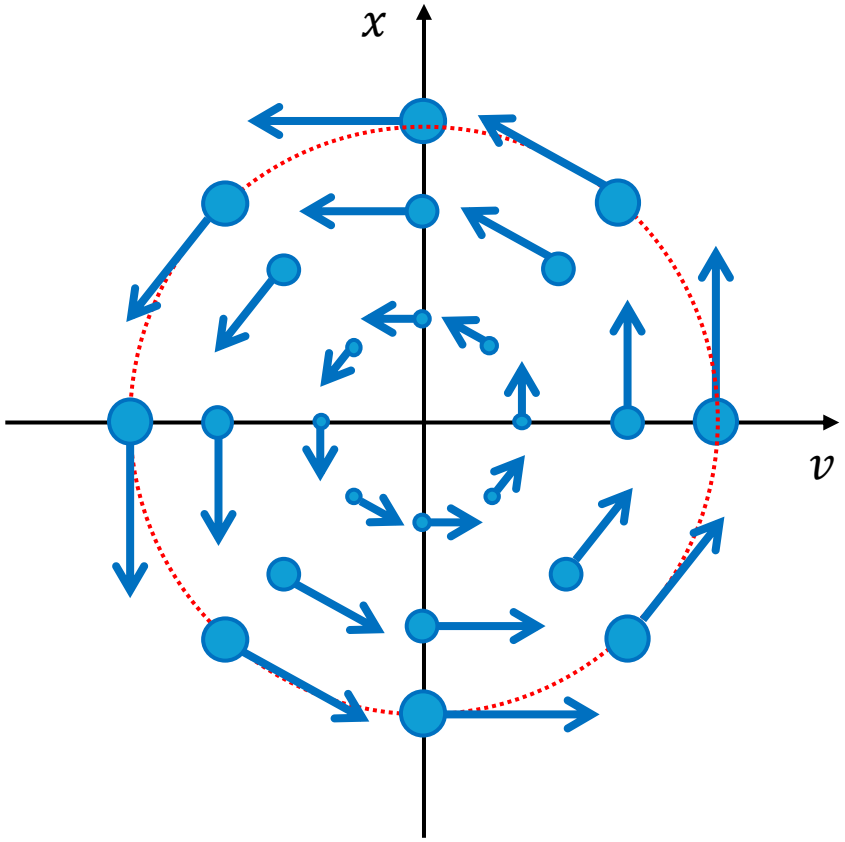
We can't always solve this equation analytically.
What are the numerical approximators?

Time Integration Algorithms

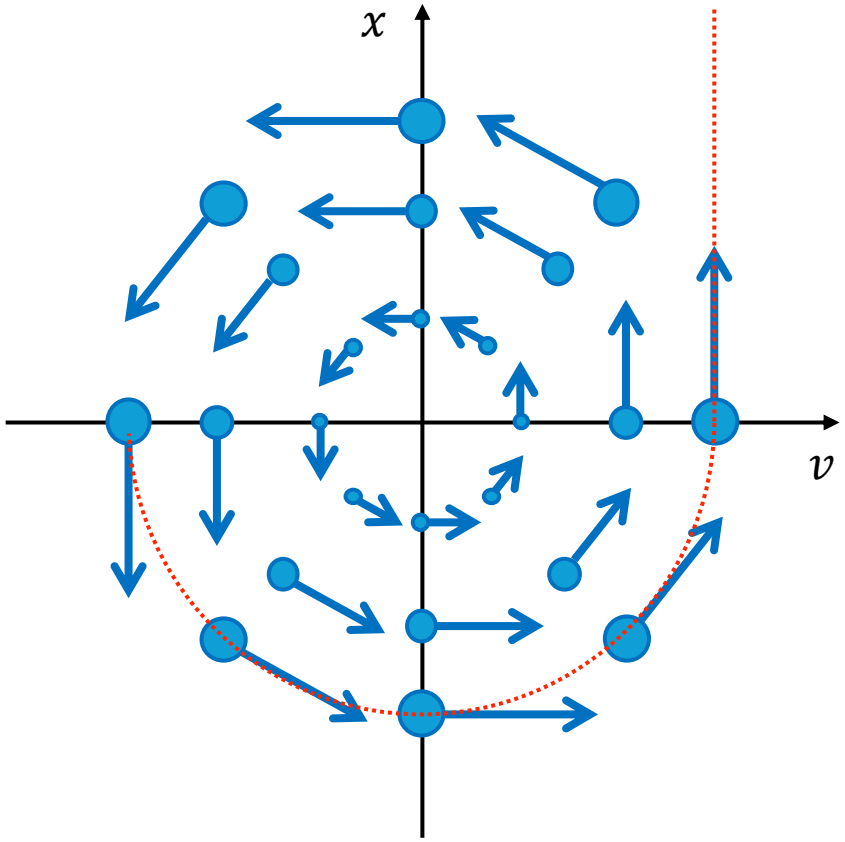
- Problem: How to solve the ODE numerically? We can only approximate infinitesimal dt with discrete Δt
- Two types of time integration algorithms:
  1. Explicit time integration methods: Next time step can be computed entirely using values from the current time step or before
  2. Implicit time integration methods: Next time step is computed using values from the future
- How to choose the time integration algorithms?
  - Performance
  - Stability
  - Accuracy

Slide adapted from D. Levin

Stability of Time Integration Algorithms



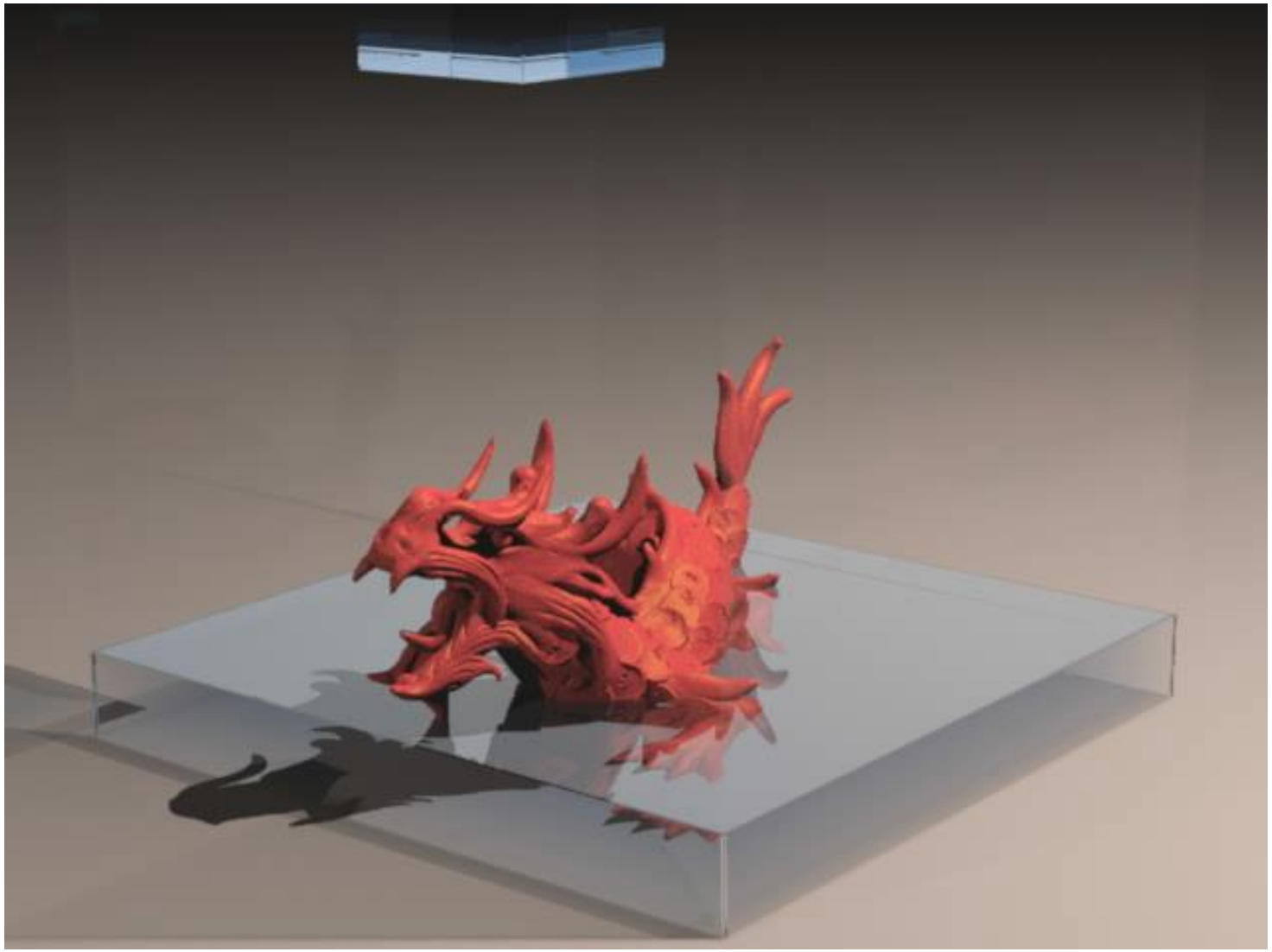
Stable: the path traced out by the numerical approximator stays the same as the analytical solution



Unstable: the path traced out by the numerical approximator diverges from the analytical solution

Slide adapted from D. Levin

Accuracy of Time Integration Algorithms



Slide adapted from D. Levin

21 A parallel multigrid Poisson solver for fluids simulation on large grids. McAdams et al.

Forward Euler Time Integration

- Our ODE equation:

$$\underbrace{\begin{pmatrix} m & 0 \\ 0 & 1 \end{pmatrix}}_{A} \underbrace{\frac{d}{dt}\begin{pmatrix} v \\ x \end{pmatrix}}_{\dot{y}} = \underbrace{\begin{pmatrix} 0 & -k \\ 1 & 0 \end{pmatrix}\overbrace{\begin{pmatrix} v \\ x \end{pmatrix}}^{y}}_{f(y)}$$

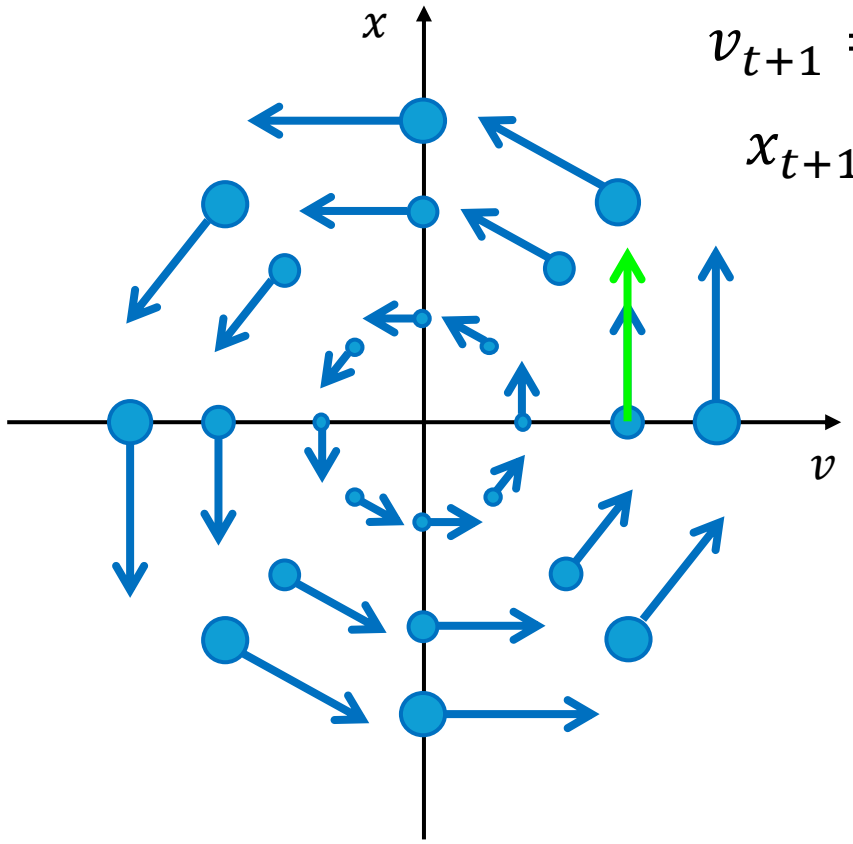
- We have: $\dot{y} \approx \frac{1}{\Delta t}(y_{t+1} - y_t)$
- Substitute $\dot{y}$ in the original ODE:

$$A\frac{1}{\Delta t}(y_{t+1} - y_t) = f(y_t) \Rightarrow \underbrace{y_{t+1} = \frac{\Delta t}{A} f(y_t) + y_t}$$

$$v_{t+1} = v_t - \Delta t \frac{k}{m} x_t$$

$$x_{t+1} = x_t + \Delta t v_t$$

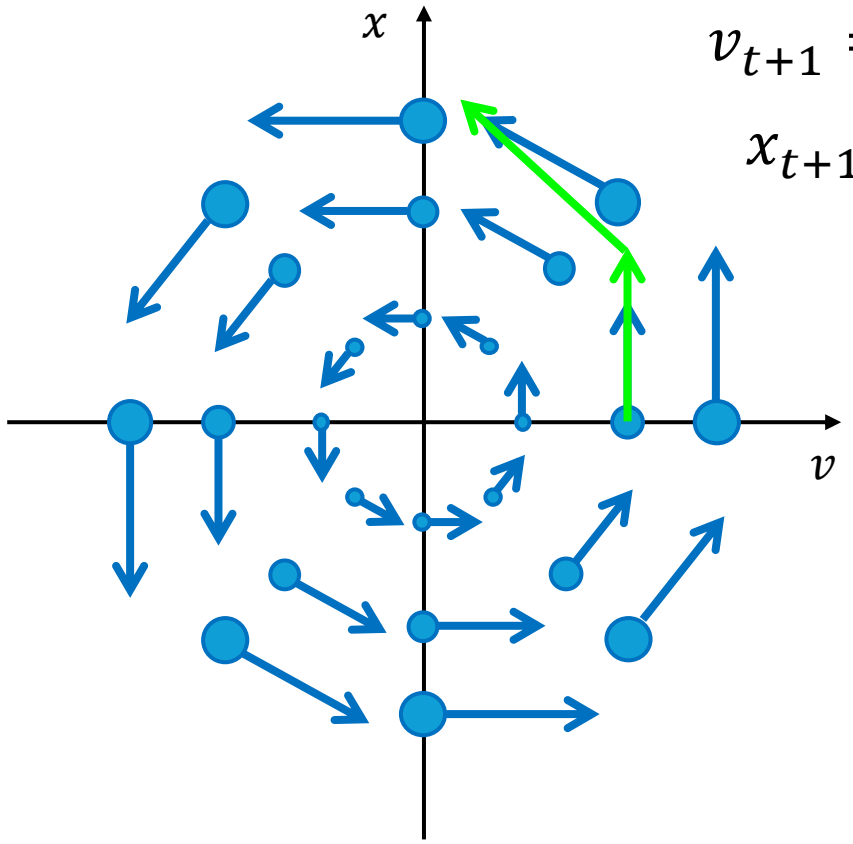
Forward Euler Time Integration



$$v_{t+1} = v_t - \Delta t \frac{k}{m} x_t$$

$$x_{t+1} = x_t + \Delta t v_t$$

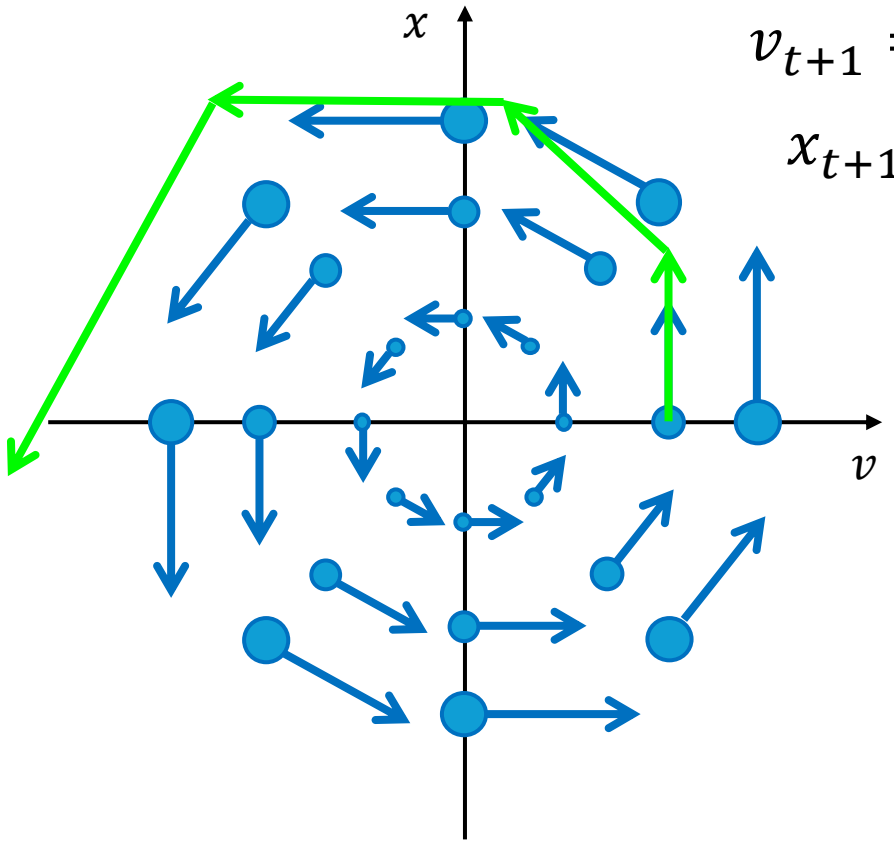
Forward Euler Time Integration



$$v_{t+1} = v_t - \Delta t \frac{k}{m} x_t$$

$$x_{t+1} = x_t + \Delta t v_t$$

Forward Euler Time Integration



$$v_{t+1} = v_t - \Delta t \frac{k}{m} x_t$$

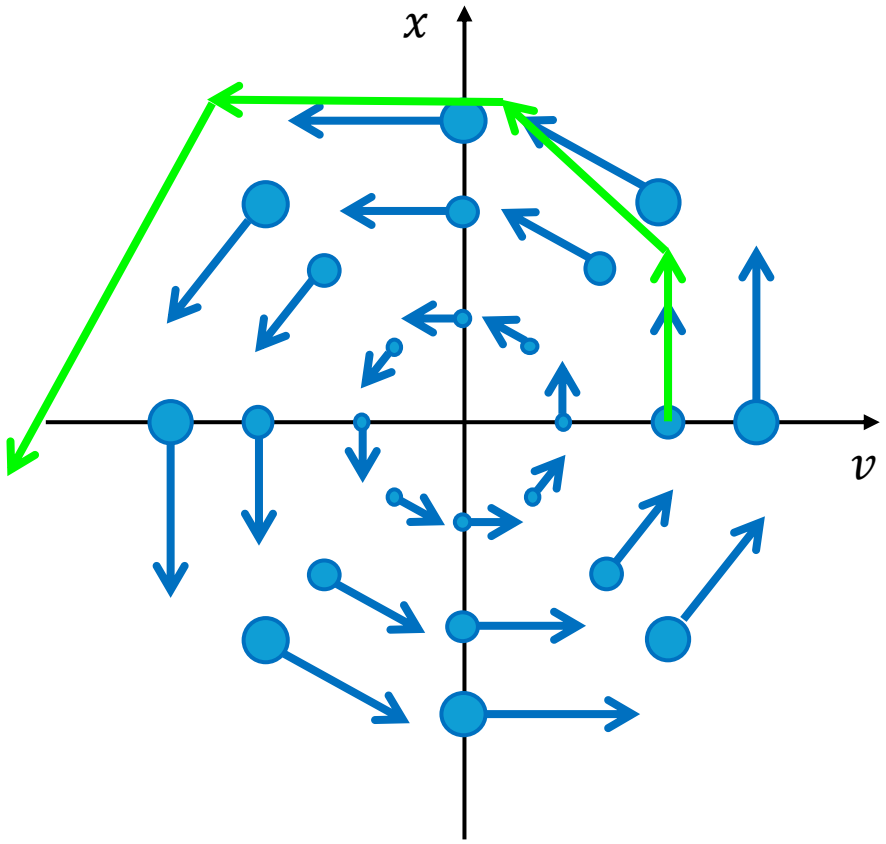
$$x_{t+1} = x_t + \Delta t v_t$$

Forward Euler Time Integration

- Forward Euler time integration updates y with "current slope":

$$y_{t+1} = \frac{\Delta t}{A} f(y_t) + y_t$$

- Problem: the integrated values overshoot as the numerical update lags behind the analytical solution
- Idea:
  - Update y with "average slope" of the current and future slope.



Runge-Kutta Time Integration

- Update y with "average slope" with real (current/previous) slope $f(y_{t+a})$ and estimated (future) slope $f(\tilde{y}_{t+b})$:

averaging coefficients

$$y_{t+1} = y_t + \frac{\Delta t}{A}\left(\alpha f(y_{t+a}) + \beta f(\tilde{y}_{t+b})\right)$$

- Using Forward Euler Time Integration to estimate $\tilde{y}_{t+b}$:

$$\tilde{y}_{t+b} = \frac{b\Delta t}{A} f(y_t) + y_t$$

Slide adapted from D. Levin

Heun's Method: A Special Case of Runge-Kutta Time Integration

- Update y with "average slope" with the current slope $f(y_t)$ and estimated future slope $f(\tilde{y}_{t+1})$:

$$y_{t+1} = y_t + \frac{\Delta t}{A}\left(\frac{1}{2}f(y_t) + \frac{1}{2}f(\tilde{y}_{t+1})\right)$$

- Using Forward Euler Time Integration to estimate $\tilde{y}_{t+1}$:

$$\tilde{y}_{t+1} = \frac{\Delta t}{A}f(y_t) + y_t$$

Slide adapted from D. Levin

Generalization of Runge-Kutta Time Integration

$$\kappa_1 = \mathbf{A}^{-1}\mathbf{f}\left(\mathbf{y}^t\right)$$

$$\kappa_2 = \mathbf{A}^{-1}\mathbf{f}\left(\mathbf{y}^t + \Delta t \cdot \kappa_1\right)$$

$$\mathbf{y}^{t+1} = \mathbf{y}^t + \frac{\Delta t}{2}\left(\kappa_1 + \kappa_2\right)$$

Slide adapted from D. Levin

Generalization of Runge-Kutta Time Integration

Fourth-Order Runge Kutta

$$\kappa_1 = \mathrm{A}^{-1}\mathbf{f}\left(\mathbf{y}^t\right)$$

$$\kappa_2 = \mathrm{A}^{-1}\mathbf{f}\left(\mathbf{y}^t + \frac{\Delta t}{2}\cdot\kappa_1\right)$$

$$\kappa_3 = \mathrm{A}^{-1}\mathbf{f}\left(\mathbf{y}^t + \frac{\Delta t}{2}\cdot\kappa_2\right)$$

$$\kappa_4 = \mathrm{A}^{-1}\mathbf{f}\left(\mathbf{y}^t + \Delta t\cdot\kappa_3\right)$$

$$\mathbf{y}^{t+1} = \mathbf{y}^t + \frac{\Delta t}{6}\left(\kappa_1 + 2\cdot\kappa_2 + 2\cdot\kappa_3 + \kappa_4\right)$$

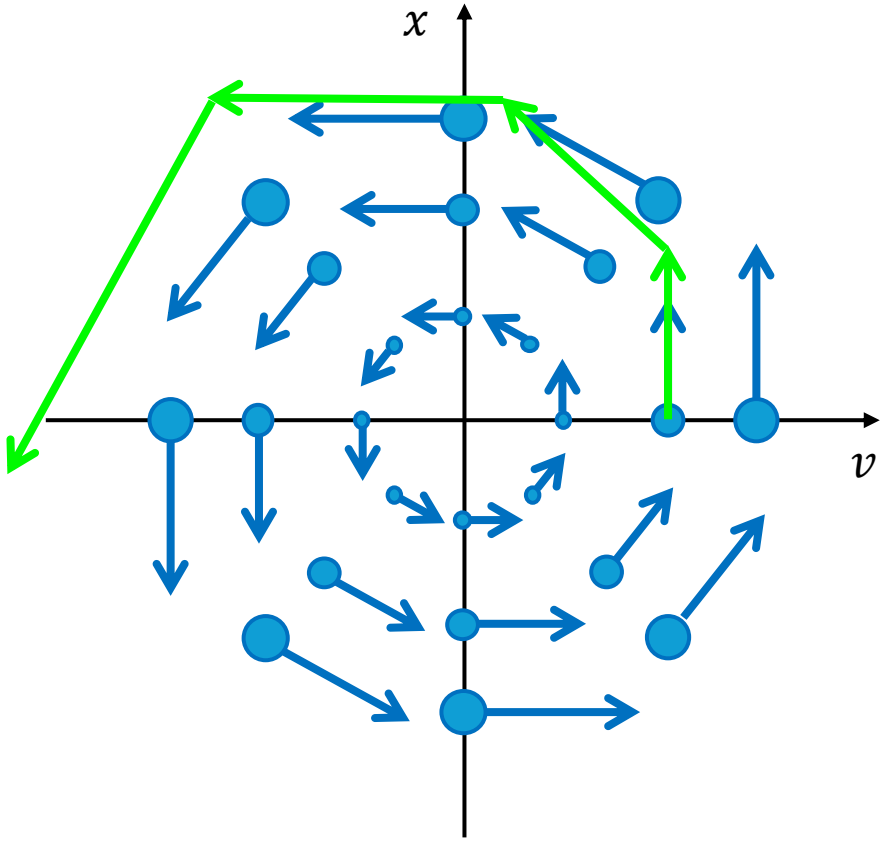
Slide adapted from D. Levin

Forward Euler Time Integration

- Forward Euler time integration updates y with "current slope":

$$y_{t+1} = \frac{\Delta t}{A} f(y_t) + y_t$$

- Problem: the integrated values overshoot as the numerical update lags behind the analytical solution
- Idea:
  - Update y with "average slope" of the current and future slope.
  - Update y with "future slope"



Backward Euler Time Integration

- Our ODE equation:

$$\underbrace{\begin{pmatrix} m & 0 \\ 0 & 1 \end{pmatrix}}_{A} \underbrace{\frac{d}{dt}\begin{pmatrix} v \\ x \end{pmatrix}}_{\dot{y}} = \underbrace{\overbrace{\begin{pmatrix} 0 & -k \\ 1 & 0 \end{pmatrix}}^{B} \overbrace{\begin{pmatrix} v \\ x \end{pmatrix}}^{y}}_{f(y)}$$

- We have: $\dot{y} \approx \frac{1}{\Delta t}(y_{t+1} - y_t)$
- The original ODE can be re-written as:

$$A\frac{1}{\Delta t}(y_{t+1} - y_t) = f(y_{t+1})$$

Update y with future slope

Backward Euler Time Integration

- Our ODE equation:

$$\underbrace{\begin{pmatrix} m & 0 \\ 0 & 1 \end{pmatrix}}_{A} \underbrace{\frac{d}{dt}\begin{pmatrix} v \\ x \end{pmatrix}}_{\dot{y}} = \underbrace{\boxed{\begin{pmatrix} 0 & -k \\ 1 & 0 \end{pmatrix}}}_{B} \underbrace{\boxed{\begin{pmatrix} v \\ x \end{pmatrix}}}_{y} \Rightarrow \underbrace{\quad\quad}_{f(y)}$$

- We have: $\dot{y} \approx \frac{1}{\Delta t}(y_{t+1} - y_t)$
- The original ODE can be re-written as:

$$A\frac{1}{\Delta t}(y_{t+1} - y_t) = f(y_{t+1}) \Rightarrow y_{t+1} = \frac{\Delta t}{A} B y_{t+1} + y_t$$

$$\Rightarrow \left(I - \frac{\Delta t}{A} B\right) y_{t+1} = y_t$$

Slide adapted from D. Levin

Backward Euler Time Integration

- How to solve $\left(I - \frac{\Delta t}{A} B\right) y_{t+1} = y_t$?
- We have:

$$v_{t+1} + \Delta t \frac{k}{m} x_{t+1} = v_t$$

$$x_{t+1} - \Delta t v_{t+1} = x_t$$

- Substitute x_{t+1} into the first equation, we have:

$$v_{t+1} + \Delta t \frac{k}{m} (x_t + \Delta t v_{t+1}) = v_t$$

$$\Rightarrow \left(1 + \Delta t^2 \frac{k}{m}\right) v_{t+1} = v_t - \Delta t \frac{k}{m} x_t$$

Slide adapted from D. Levin

Backward Euler Time Integration

- How to solve $\left(I - \frac{\Delta t}{A} B\right) y_{t+1} = y_t$?
- We have:

$$v_{t+1} + \Delta t \frac{k}{m} x_{t+1} = v_t$$

$$\boxed{x_{t+1} - \Delta t v_{t+1} = x_t}$$

- Substitute x_{t+1} into the first equation, we have:

$$v_{t+1} + \Delta t \frac{k}{m} (x_t + \Delta t v_{t+1}) = v_t$$

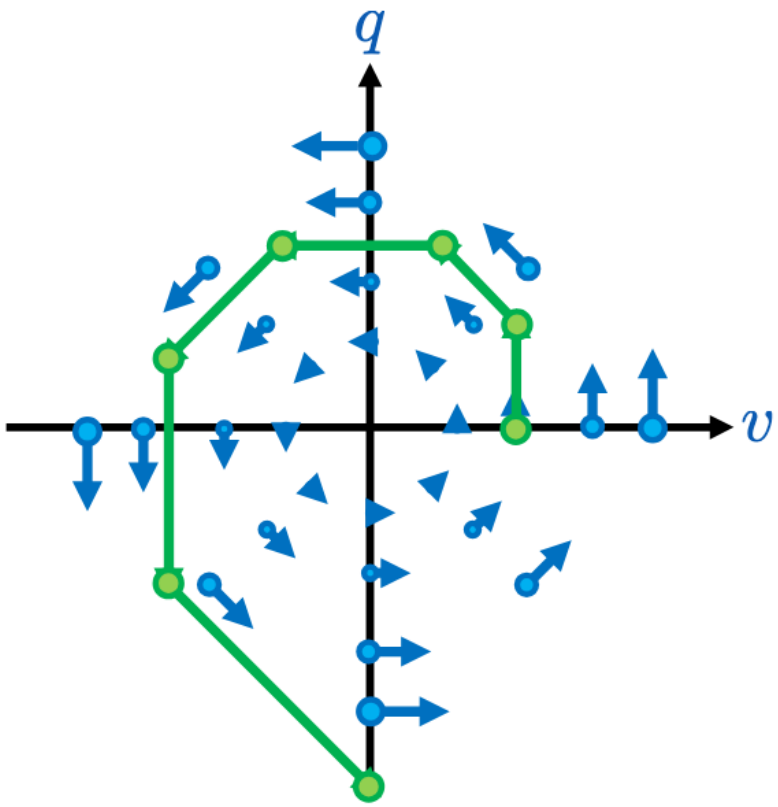
$$\Rightarrow \boxed{\left(1 + \Delta t^2 \frac{k}{m}\right) v_{t+1} = v_t - \Delta t \frac{k}{m} x_t}$$

We solve the backward integration implicitly!

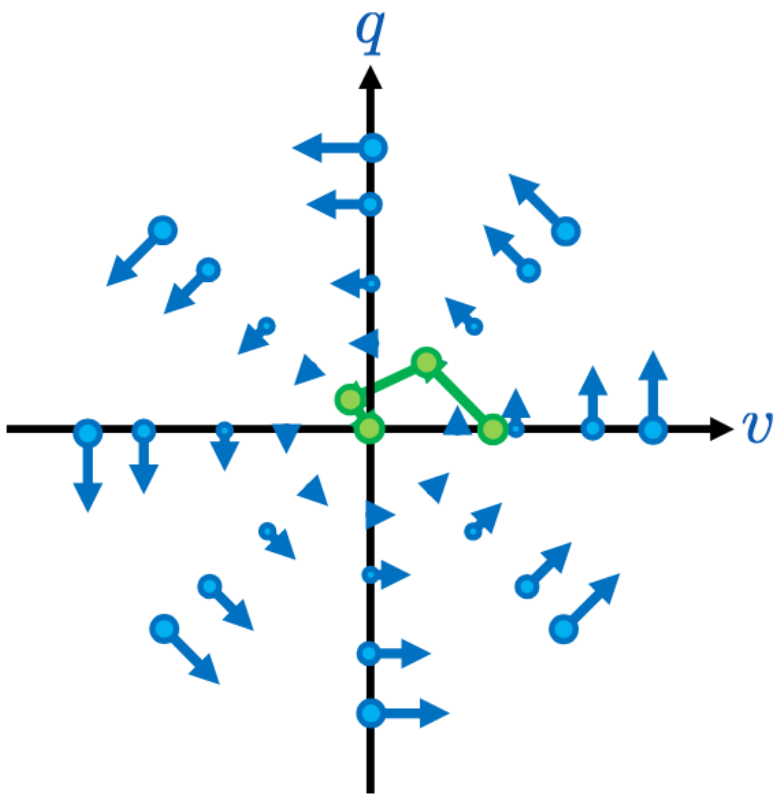
Slide adapted from D. Levin

Forward vs. Backward Euler Time Integration

- Can we cancel out the exploding and damping effect? Yes! Let's try to mix these two integration methods.



Explicit (exploding!!!)



Implicit (damping)

Slide credit D. Levin

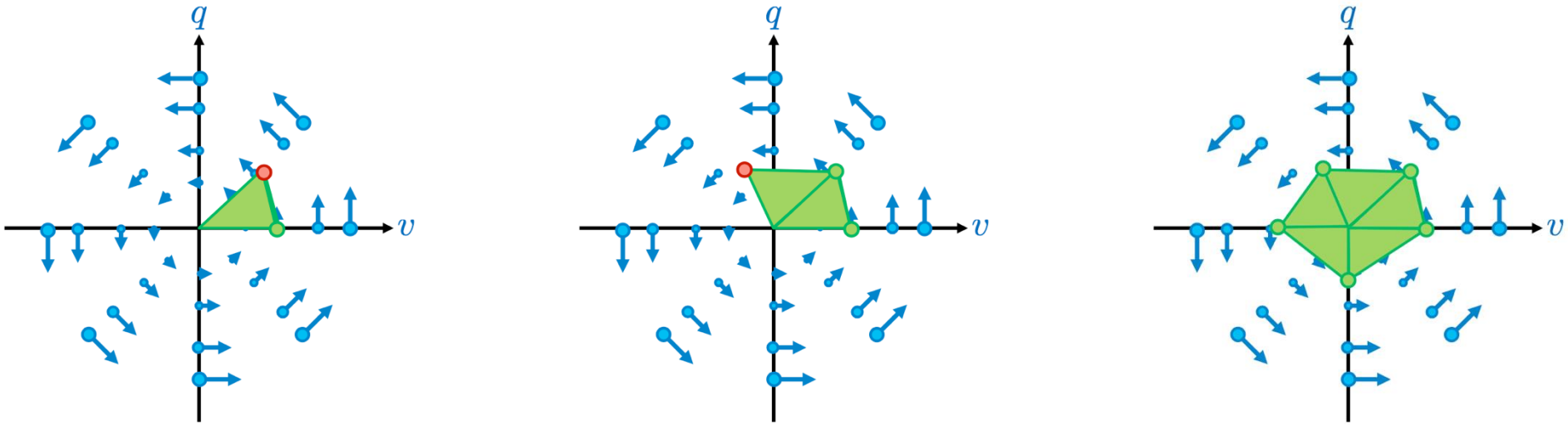
Symplectic Euler Time Integration

- First, take an explicit step

$$v_{t+1} = v_t - \Delta t \frac{k}{m} x_t$$

- Next, take an implicit step

$$x_{t+1} = x_t - \Delta t v_{t+1}$$

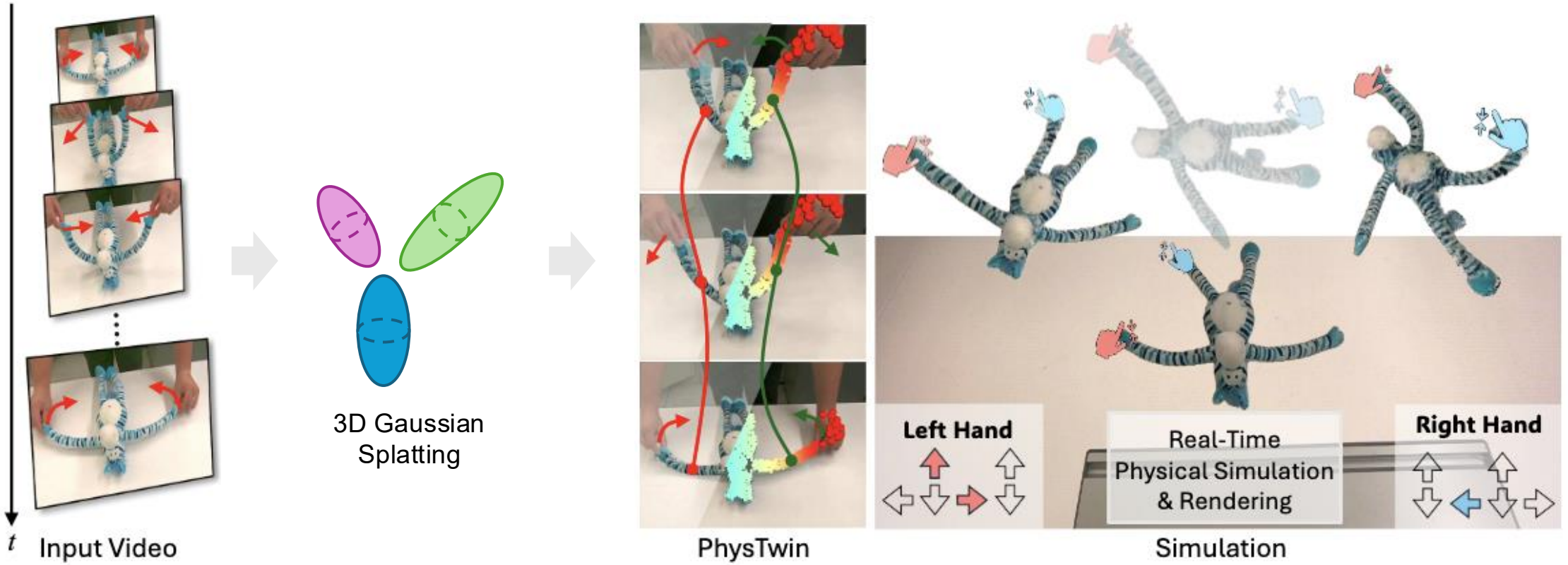


Slide adapted from D. Levin

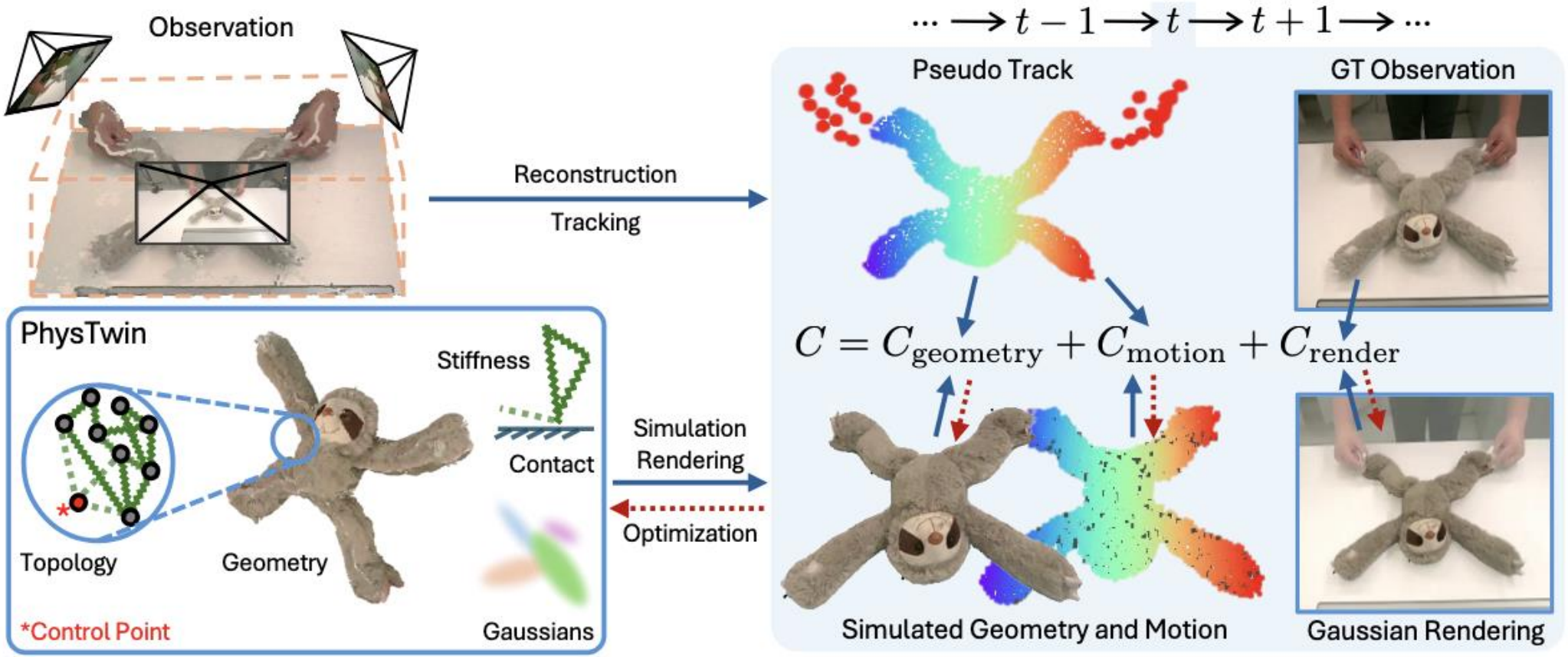
Scene Reconstruction and Simulation with Mass-Spring System

PhysTwin: Physics-Informed Reconstruction and Simulation of Deformable Objects from Videos. Jiang et al.

Idea: Integrate 3D Gaussian Splatting with Mass-Spring Modeling



PhysTwin: Physics-Informed Reconstruction and Simulation of Deformable Objects from Videos

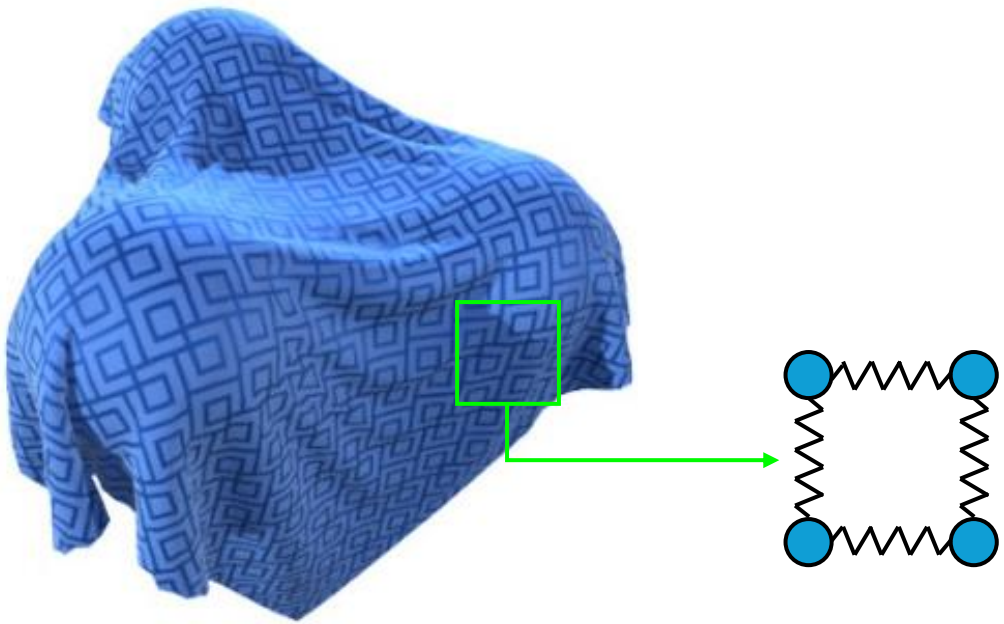


Results: Real-to-Sim Demo

Content

- Non-Rigid Modeling
  - ➢Mass-spring system
  - ➢Position-based Dynamics
  - ➢Tetrahedral and Deformation

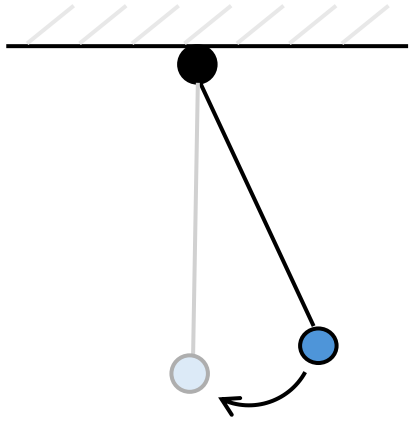
Mass-Spring System Has Problems

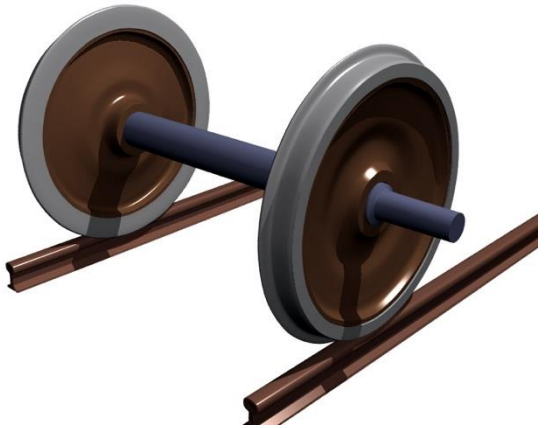


- How to describe objects of different kinetic constraints, e.g. bending, impenetrability etc?
- How to model continuum material? The empty space between spring-connected nodes has no mass and energy.

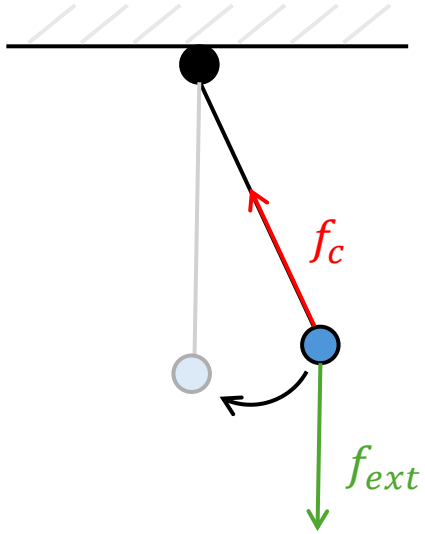
A Dynamic System May Have Multiple Constraints

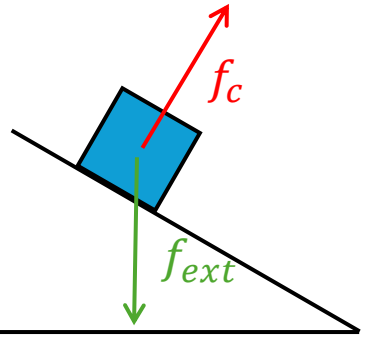
- Constraints limit the motion of objects by restricting them to follow a specific trajectory or path. For example:
  - The circular motion of a pendulum
  - The distance constraints between two objects
  - The resistance when bending an iron rod





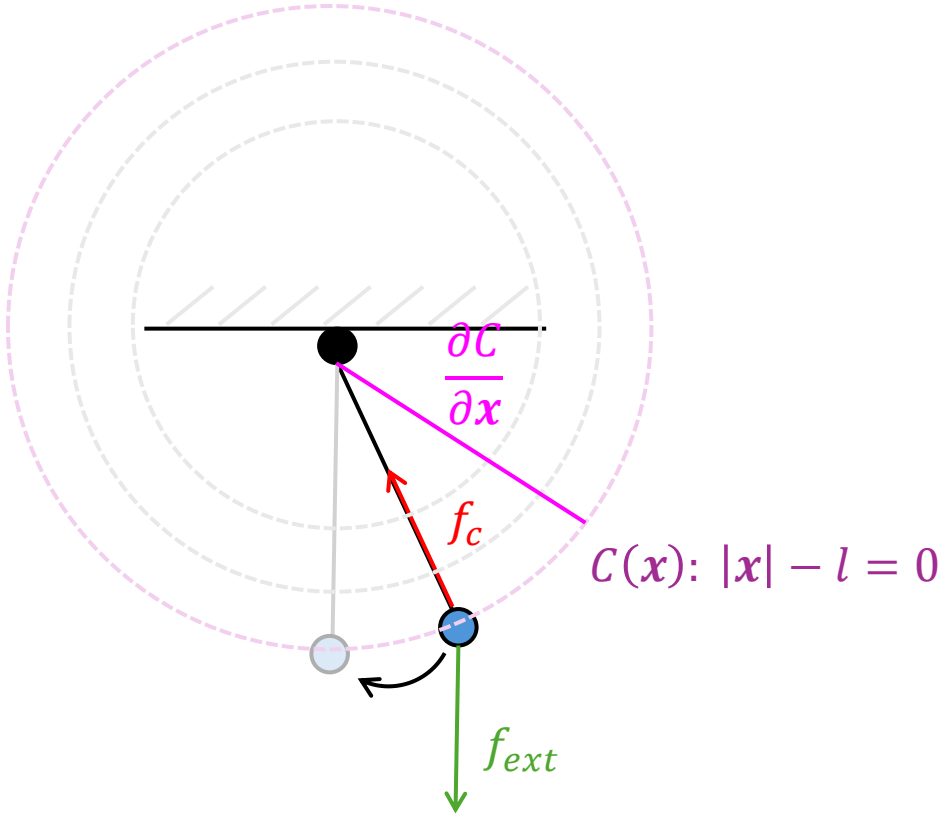
How to Solve a Dynamic System with Constraints?





- Idea 1: Directly calculate the forces required to maintain the constraints
- The job of these constraint forces is to cancel just those parts of the applied forces that act against the constraints

How to Solve a Dynamic System with Constraints?



- Idea 1: Directly calculate the forces required to maintain the constraints
- The job of these constraint forces is to cancel just those parts of the applied forces that act against the constraints
- For a constraint C, the corresponding constraint force is $f_c = \lambda \frac{\partial C}{\partial x}$ where x denotes the spatial coordinates of the dynamic system

Check "Physically Based Modeling: Principles and Practice Constrained Dynamics by Andrew Witkin" for more details

How to Solve a Dynamic System with Constraints?

- Idea 2: Calculate the kinetic update with Newton's law and refine the kinetic update to follow the constraints

- Problem formulation: Given a set of positions $\boldsymbol{x} = [x_1, \cdots, x_N]$ and constraints $\boldsymbol{C_1(x)}, \cdots, \boldsymbol{C_M(x)}$, the goal is to find new positions $\boldsymbol{p} = [p_1, \cdots, p_N]$ such that $\boldsymbol{C_i(p)} \geq 0 \; \forall i$

Position Based Dynamics

Algorithm 1 Position-based dynamics

1: **for all** vertices i **do**
2: initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_i^0$, $w_i = 1/m_i$
3: **end for**
4: **loop**
5: **for all** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$
6: **for all** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
7: **for all** vertices i **do** genCollConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
8: **loop** solverIteration **times**
9: projectConstraints($C_1, \ldots, C_{M+M_{\text{Coll}}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
10: **end loop**
11: **for all** vertices i **do**
12: $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
13: $\mathbf{x}_i \leftarrow \mathbf{p}_i$
14: **end for**
15: velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
16: **end loop**

Derive the initial kinetic update with Newton's law

Modify the kinetic update based on constraints

A Survey on Position Based Dynamics, 2017. Bender et al.
Position Based Dynamics. Müller et al.

Position Based Dynamics

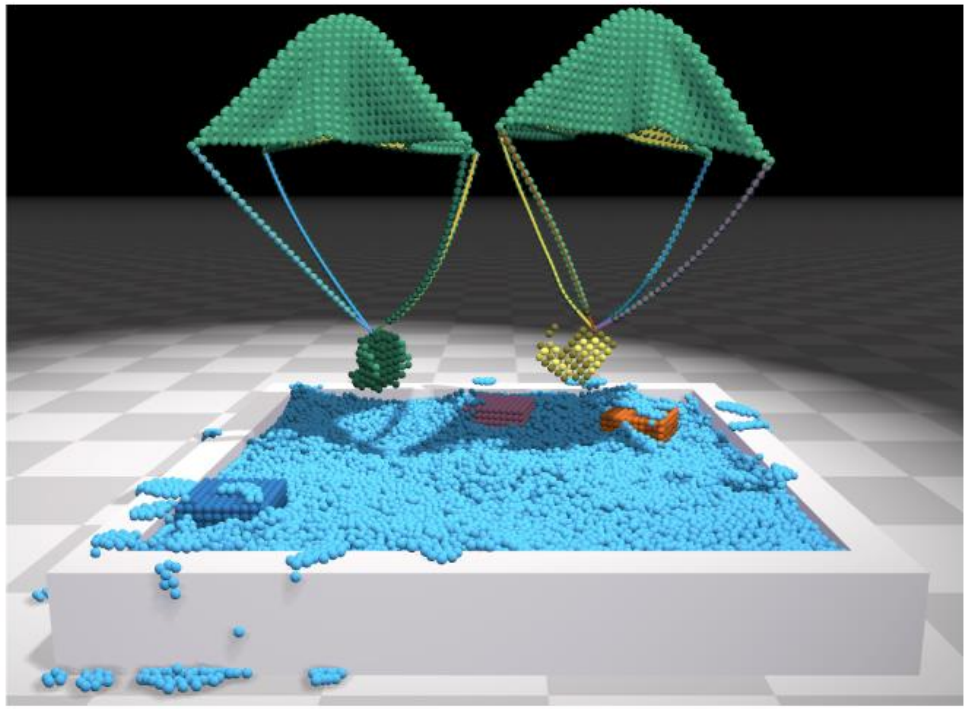
Algorithm 1 Position-based dynamics

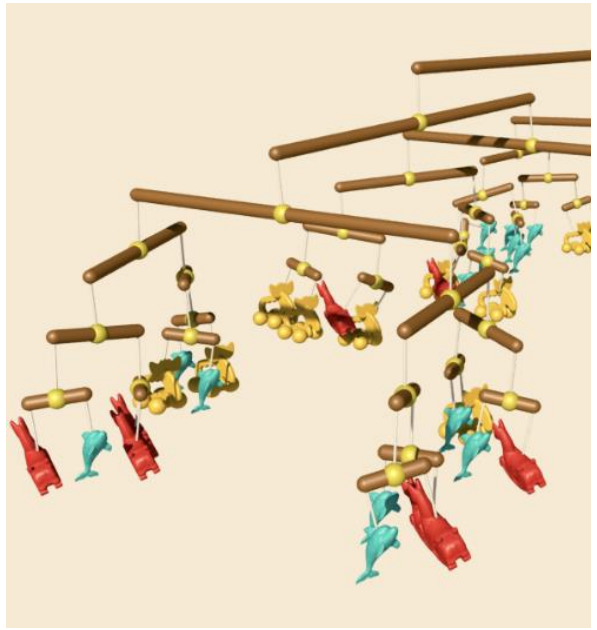
1: **for all** vertices i **do**
2: initialize $\mathbf{x}_i = \mathbf{x}_i^0$, $\mathbf{v}_i = \mathbf{v}_i^0$, $w_i = 1/m_i$
3: **end for**
4: **loop**
5: **for all** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{\text{ext}}(\mathbf{x}_i)$
6: **for all** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$
7: **for all** vertices i **do** genCollConstraints($\mathbf{x}_i \rightarrow \mathbf{p}_i$)
8: **loop** solverIteration **times**
9: projectConstraints($C_1, \ldots, C_{M+M_{\text{Coll}}}, \mathbf{p}_1, \ldots, \mathbf{p}_N$)
10: **end loop**
11: **for all** vertices i **do**
12: $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$
13: $\mathbf{x}_i \leftarrow \mathbf{p}_i$
14: **end for**
15: velocityUpdate($\mathbf{v}_1, \ldots, \mathbf{v}_N$)
16: **end loop**

Derive the initial kinetic update with Newton's law

Modify the kinetic update based on constraints

The idea is applicable to both particles and rigid bodies





A Survey on Position Based Dynamics, 2017. Bender et al.
Position Based Dynamics. Müller et al.

Constraint Projection

- Let $\boldsymbol{p} = \boldsymbol{x} + \Delta\boldsymbol{x}$. The problem becomes to find the correction $\Delta\boldsymbol{x}$ such that $C(\boldsymbol{x} + \Delta\boldsymbol{x}) \succ 0$.

- We can first approximate the constraint equation with Taylor expansion:

$$C(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx C(\boldsymbol{x}) + \nabla C(\boldsymbol{x}) \cdot \Delta\boldsymbol{x} \succ 0 \qquad \nabla = \left(\frac{\partial}{\partial x_1}, \cdots, \frac{\partial}{\partial x_N}\right)$$

- A possibly good solution is to restrict $\Delta\boldsymbol{x}$ to be in the direction of $\nabla C(\boldsymbol{x})$. Let $M = diag(m_1, \cdots, m_N)$ be the diagonal matrix of point masses

$$\Delta\boldsymbol{x} = \lambda M^{-1} \nabla C(\boldsymbol{x})^{\mathsf{T}}$$

Why this is a possibly good solution?

Constraint Projection

- We can first approximate the constraint equation with Taylor expansion:

$$C(\boldsymbol{x} + \Delta\boldsymbol{x}) \approx C(\boldsymbol{x}) + \nabla C(\boldsymbol{x}) \cdot \Delta\boldsymbol{x} \succ 0$$

- A possibly good solution is to restrict $\Delta\boldsymbol{x}$ to be in the direction of $\nabla C(\boldsymbol{x})$.
  Let $M = diag(m_1, \cdots, m_N)$ be the diagonal matrix of point masses

$$\Delta\boldsymbol{x} = \lambda M^{-1} \nabla C(\boldsymbol{x})^{\mathsf{T}}$$

1. Linear / angular momentum is conserved: No position / velocity change along the constrained motion direction
2. The value of $C(\boldsymbol{x} + \Delta\boldsymbol{x})$ will increase

Constraint Projection

- A possibly good solution is to restrict $\boldsymbol{\Delta x}$ to be in the direction of $\boldsymbol{\nabla C(x)}$. Let $\boldsymbol{M = diag(m_1, \cdots, m_N)}$ be the diagonal matrix of point masses

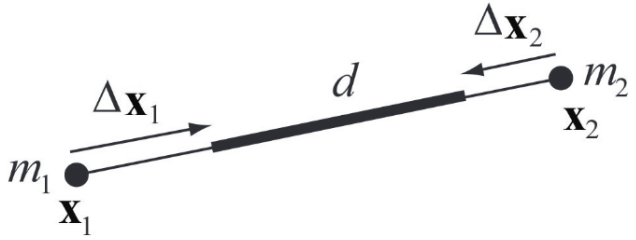
$$\boldsymbol{\Delta x = \lambda M^{-1} \nabla C(x)^{\mathsf{T}}}$$

- Let's consider the case where $\boldsymbol{C(x + \Delta x) \approx C(x) + \nabla C(x) \cdot \Delta x = 0}$, we obtain the solution of $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda = -\frac{C(x)}{\nabla C(x) M^{-1} \nabla C(x)^{\mathsf{T}}}}$$

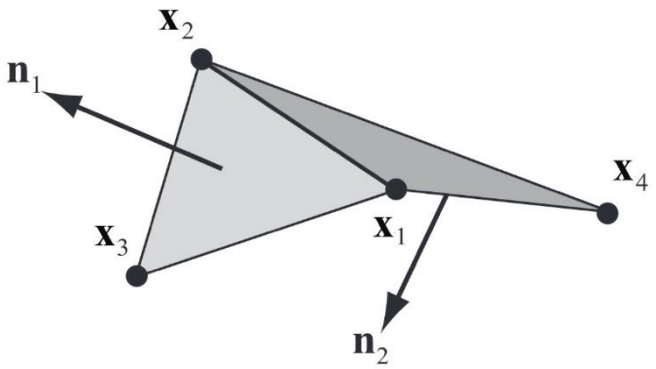
Different Types of Constraints

Stretching



$$C(x_1, x_2) = |x_1 - x_2| - d$$

Bending

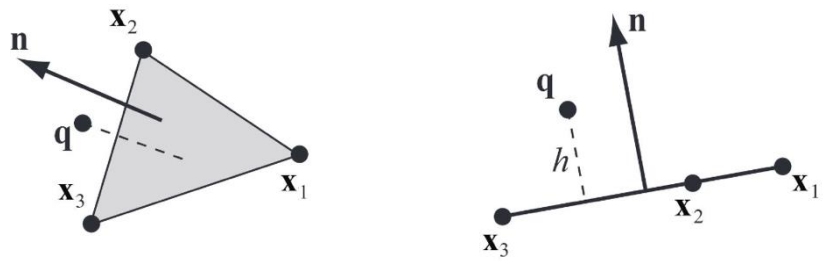


$$C_{bend}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) =$$

$$\operatorname{acos}\left(\frac{\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}}{|\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}|} \cdot \frac{\mathbf{x}_{2,1} \times \mathbf{x}_{4,1}}{|\mathbf{x}_{2,1} \times \mathbf{x}_{4,1}|}\right) - \varphi_0$$

where $x_{i,j} = x_i - x_j$ and φ_0 is the initial dihedral angle between the two triangles

Collision



$$C(\mathbf{q}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = (\mathbf{q} - \mathbf{x}_1) \cdot \frac{\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}}{|\mathbf{x}_{2,1} \times \mathbf{x}_{3,1}|} - h,$$

or

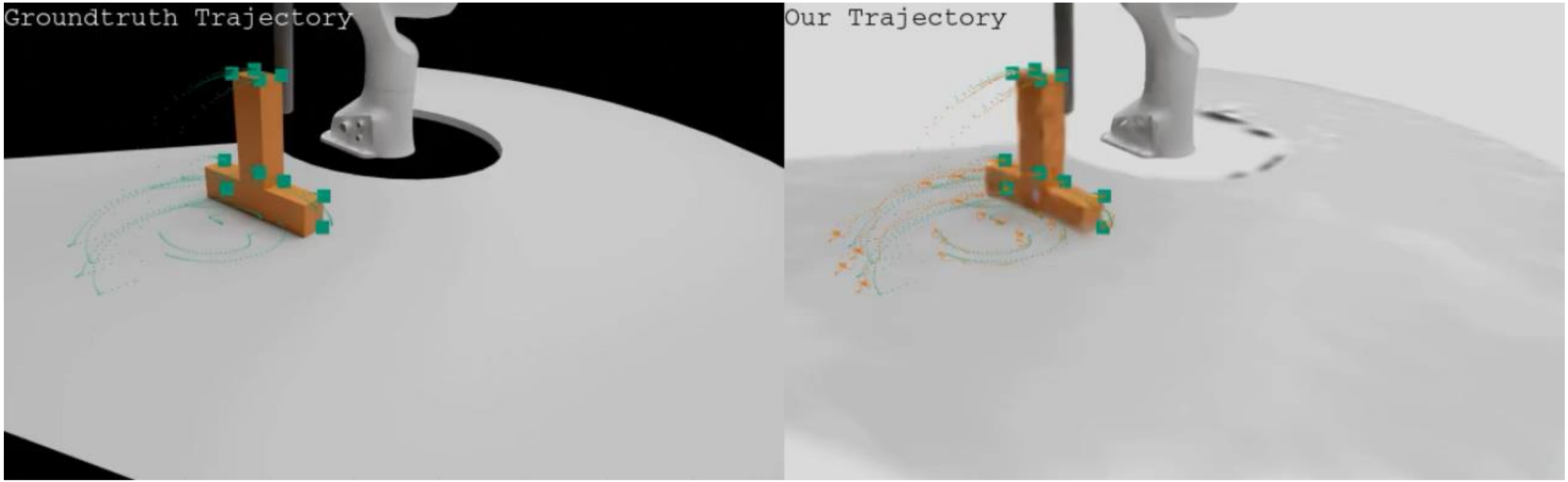
$$C(\mathbf{q}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = (\mathbf{q} - \mathbf{x}_1) \cdot \frac{\mathbf{x}_{3,1} \times \mathbf{x}_{2,1}}{|\mathbf{x}_{3,1} \times \mathbf{x}_{2,1}|} - h.$$

A Survey on Position Based Dynamics, 2017. Bender et al.
Position Based Dynamics. Müller et al.

Physical Modeling with Position Based Dynamics

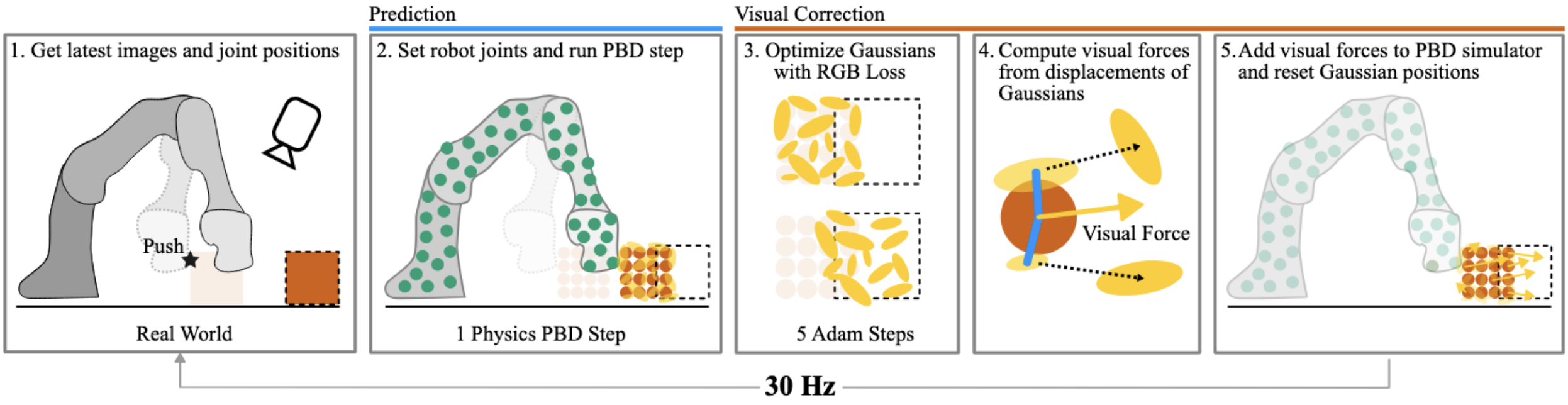
Position Based Dynamics. Müller et al.

Scene Reconstruction and Simulation with Position-Based Dynamics

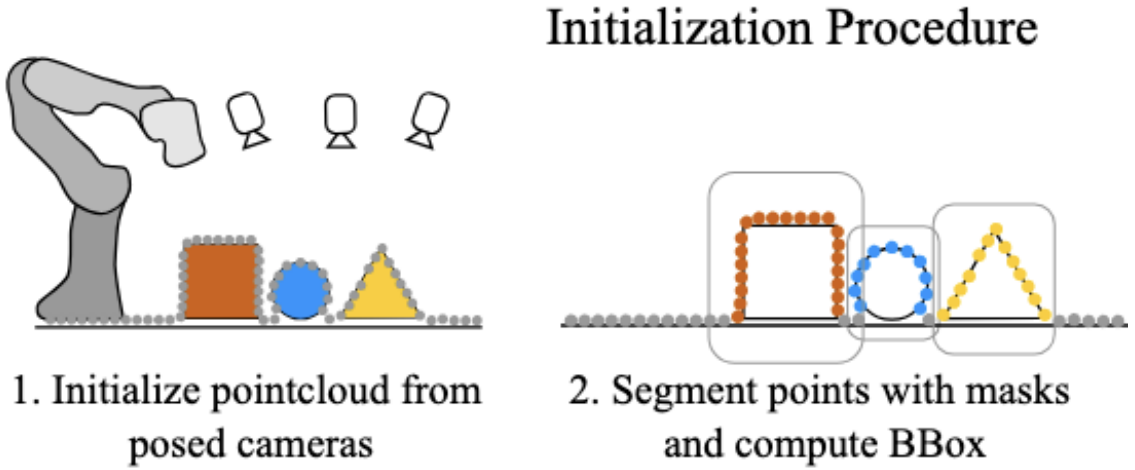


Physically Embodied Gaussian Splatting: A Visually Learnt and Physically Grounded 3D Representation for Robotics. Abou-Chakra et al

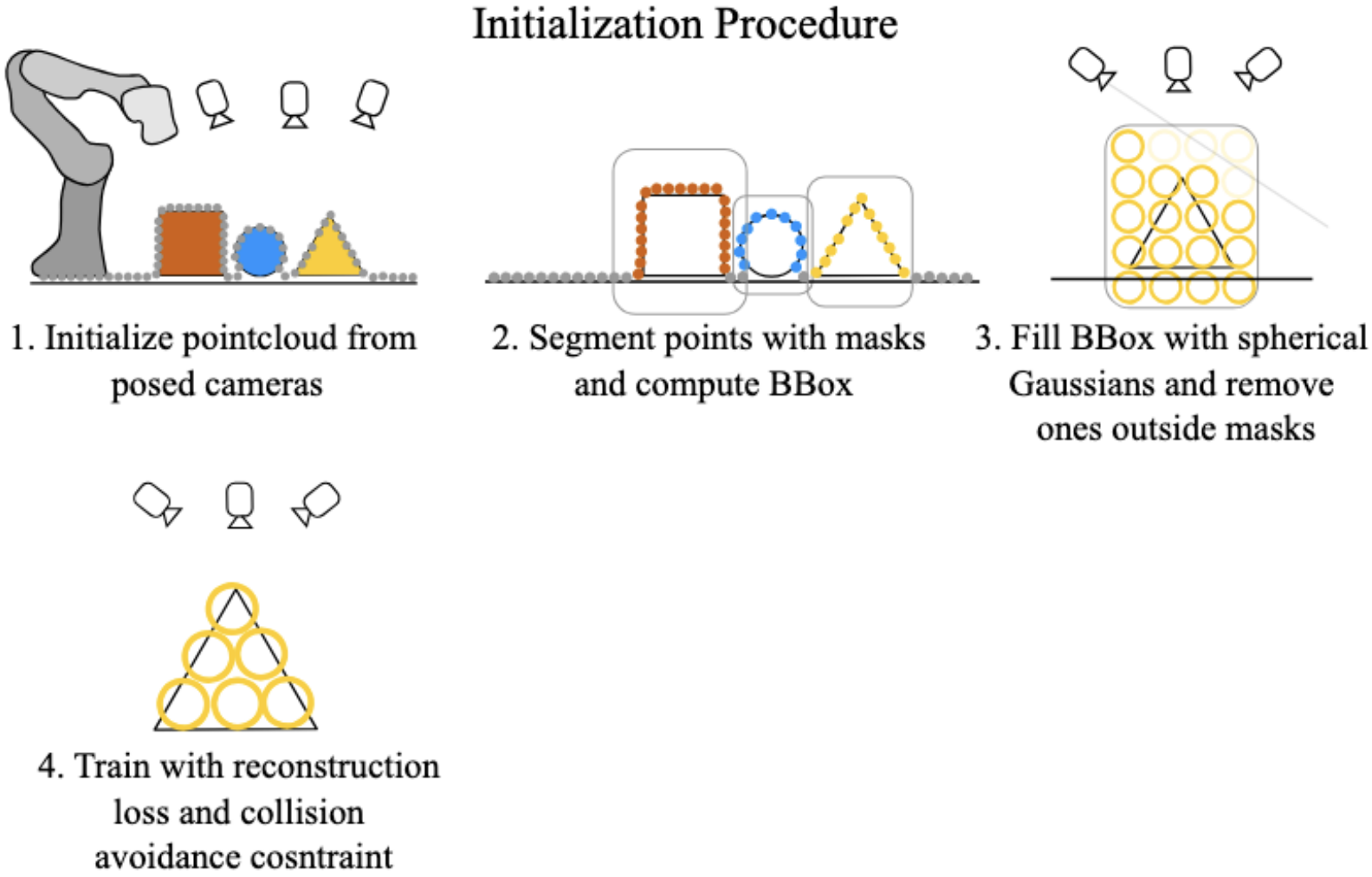
Idea: Integrate 3D Gaussian Splatting with PBD System and Correct Simulation with Visual Input



Problem: How to Couple 3D Gaussian Splatting with Particle Presentations of PBD System?

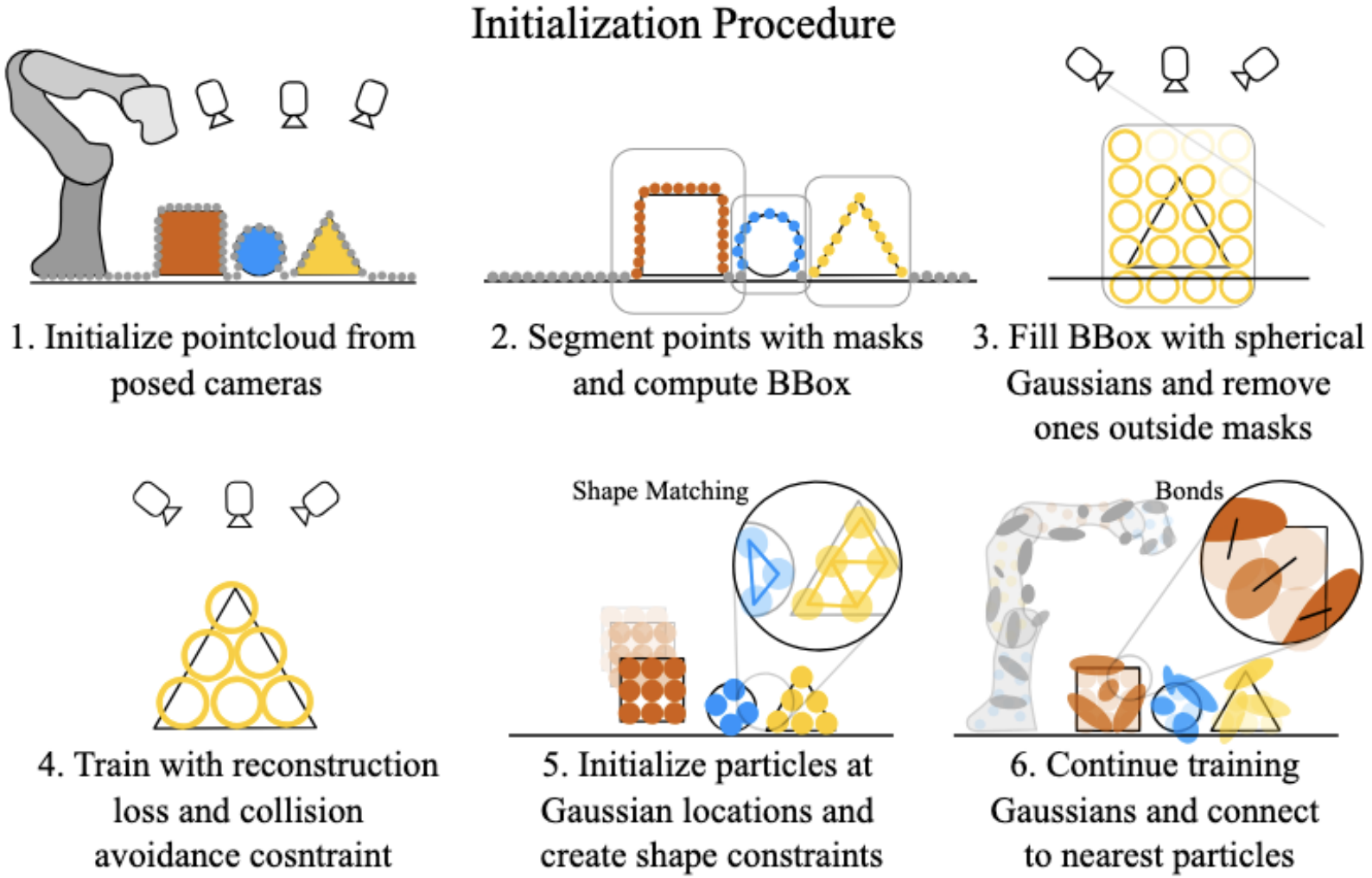


Problem: How to Couple 3D Gaussian Splatting with Particle Presentations of PBD System?

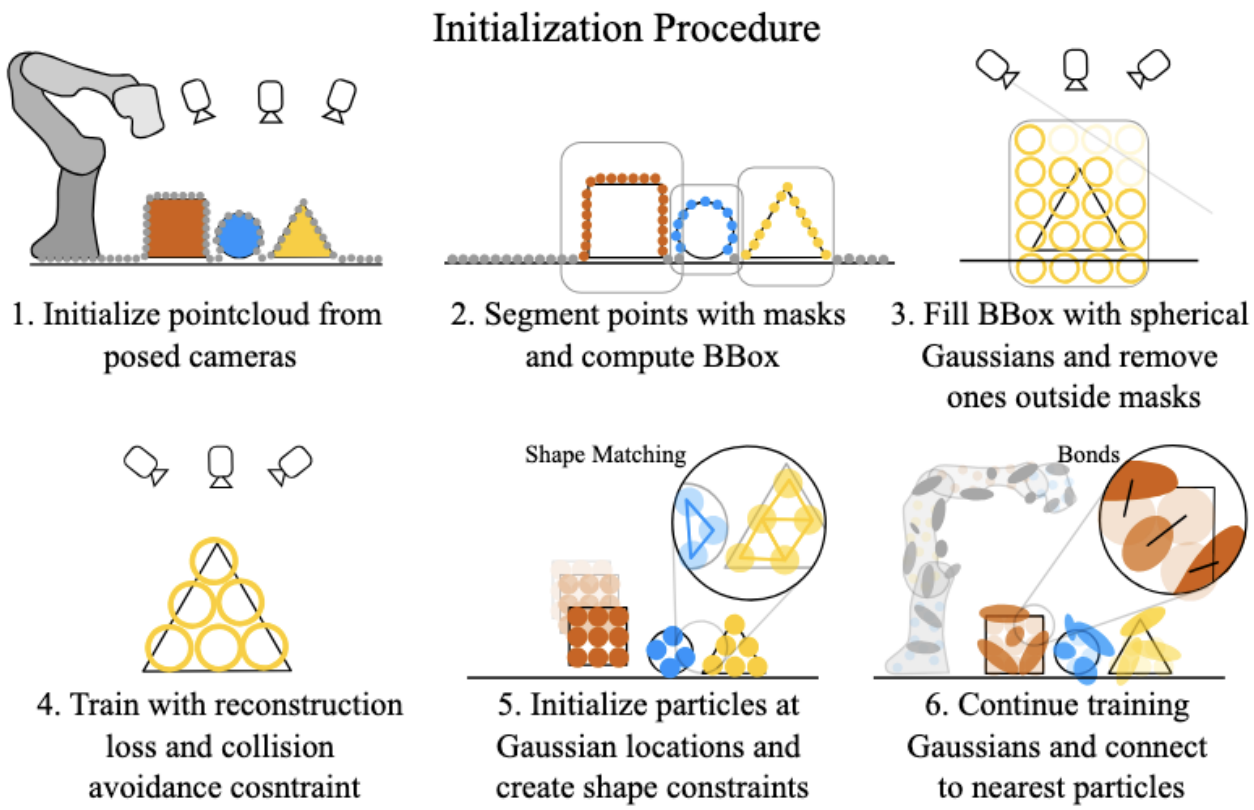


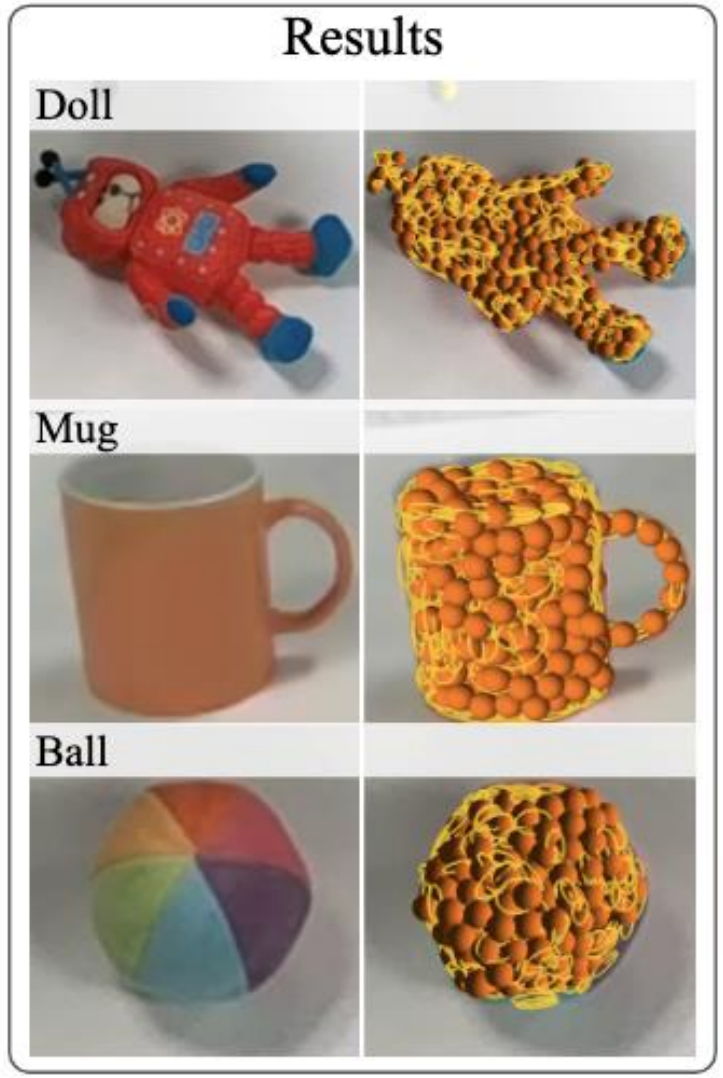
Physically Embodied Gaussian Splatting: A Visually Learnt and Physically Grounded 3D Representation for Robotics. Abou-Chakra et al

Problem: How to Couple 3D Gaussian Splatting with Particle Presentations of PBD System?



Problem: How to Couple 3D Gaussian Splatting with Particle Presentations of PBD System?





Physically Embodied Gaussian Splatting: A Visually Learnt and Physically Grounded 3D Representation for Robotics. Abou-Chakra et al

PBD Constraints

- PBD acts on oriented particles. Each particle $\boldsymbol{i}$ has position $\boldsymbol{p_i} \in \mathbb{R}^3$, velocity $\boldsymbol{v_i} \in \mathbb{R}^3$, orientation $\boldsymbol{q_i} \in \mathbb{S}^3$, angular velocity $\boldsymbol{\omega_i} \in \mathbb{R}^3$, external force $\boldsymbol{f_i} \in \mathbb{R}^3$, radius $\boldsymbol{r_i} \in \mathbb{R}^+$, and mass $\boldsymbol{m_i} \in \mathbb{R}^+$. A particle may belong to a shape $\boldsymbol{S_i}$ and thus has its resting position $\bar{\boldsymbol{p}}_{\boldsymbol{i}}$.

- Ground constraint: Prevent particles from penetrating the ground plane given by $(\boldsymbol{n}, \boldsymbol{d})$

$$\Delta \boldsymbol{p}_i^{\text{ground}} = C_{\text{ground}}(\boldsymbol{p}_i; \boldsymbol{n}, d) \cdot \boldsymbol{n}, \quad C_{\text{ground}}(\boldsymbol{p}_i; \boldsymbol{n}, d) = \min(\boldsymbol{n}^T \boldsymbol{p}_i + d - r_i, 0)$$

PBD Constraints

- PBD acts on oriented particles. Each particle i has position $p_i \in \mathbb{R}^3$, velocity $v_i \in \mathbb{R}^3$, orientation $q_i \in \mathbb{S}^3$, angular velocity $\omega_i \in \mathbb{R}^3$, external force $f_i \in \mathbb{R}^3$, radius $r_i \in \mathbb{R}^+$, and mass $m_i \in \mathbb{R}^+$. A particle may belong to a shape S_i and thus has its resting position $\bar{p}_i$.

- Particle collision constraint: Operate on a pair of particles i and j is used to model collisions

$$\Delta \boldsymbol{p}_i^{\text{col}} = \frac{w_i}{w_i + w_j} \frac{\boldsymbol{p}_i - \boldsymbol{p}_j}{||\boldsymbol{p}_i - \boldsymbol{p}_j||} C_{\text{col}(\boldsymbol{p}_i, \boldsymbol{p}_j)}, \quad C_{\text{col}}(\boldsymbol{p}_i, \boldsymbol{p}_j) = \min(||\boldsymbol{p}_i - \boldsymbol{p}_j|| - r_i - r_j, 0)$$

PBD Constraints

- PBD acts on oriented particles. Each particle i has position $p_i \in \mathbb{R}^3$, velocity $v_i \in \mathbb{R}^3$, orientation $q_i \in \mathbb{S}^3$, angular velocity $\omega_i \in \mathbb{R}^3$, external force $f_i \in \mathbb{R}^3$, radius $r_i \in \mathbb{R}^+$, and mass $m_i \in \mathbb{R}^+$. A particle may belong to a shape S_i and thus has its resting position $\bar{p}_i$.

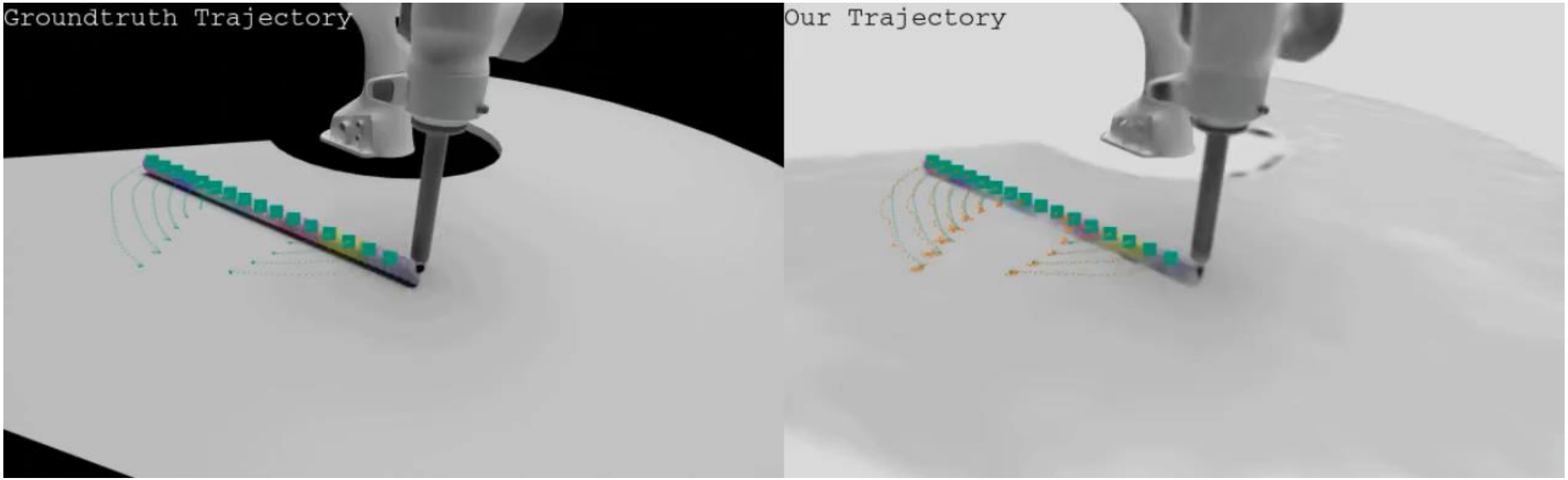
- Shape matching constraint: Ensure a group of particles belonging to a particular object (either deformable or rigid) maintain their structure throughout the simulation

$$\boxed{\boldsymbol{A}_S} = \sum_{i \in S} \frac{1}{5} m_i \boldsymbol{R}_i + \boldsymbol{p}_i \bar{\boldsymbol{p}}_i^T - M \boldsymbol{c}_S \bar{\boldsymbol{c}}_S^T, \quad \boldsymbol{c}_S = \frac{\sum_{i \in S} m_i \boldsymbol{p}_i}{M}, \bar{\boldsymbol{c}}_S = \frac{\sum_{i \in S} m_i \bar{\boldsymbol{p}}_i}{M}, M = \sum_{i \in S} m_i$$

$A_s = R_s S$ can be decomposed into a rotation matrix R_s and a symmetrical matrix S

$$\Delta \boldsymbol{p}_i^{\text{shape}} = k_S [\boldsymbol{R}_S (\bar{\boldsymbol{p}}_i - \bar{\boldsymbol{c}}_S) + \boldsymbol{c} - \boldsymbol{p}_i]$$

Results: Real-to-Sim Demo

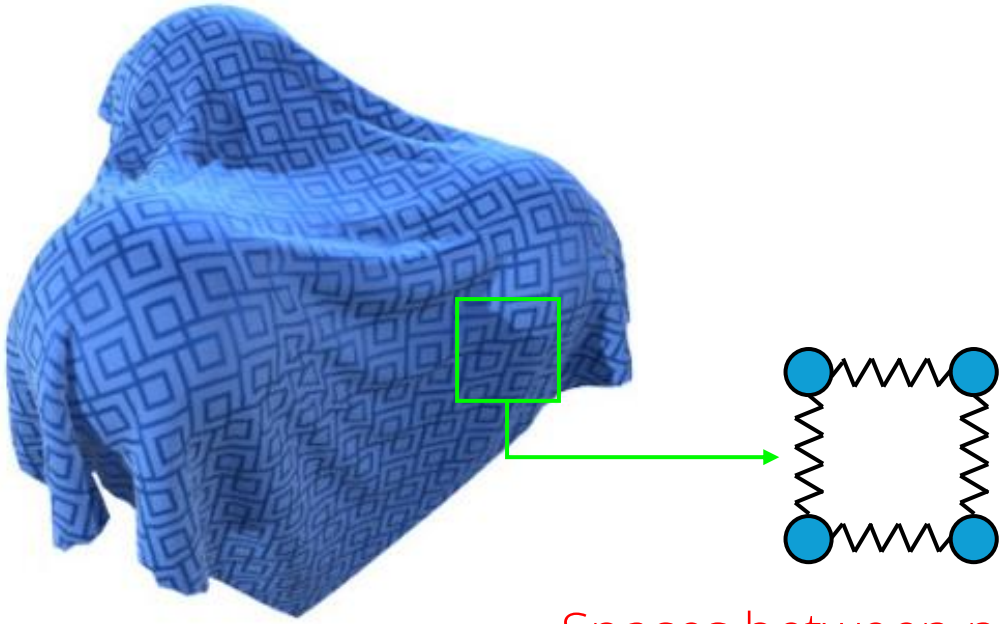


Physically Embodied Gaussian Splatting: A Visually Learnt and Physically Grounded 3D Representation for Robotics. Abou-Chakra et al

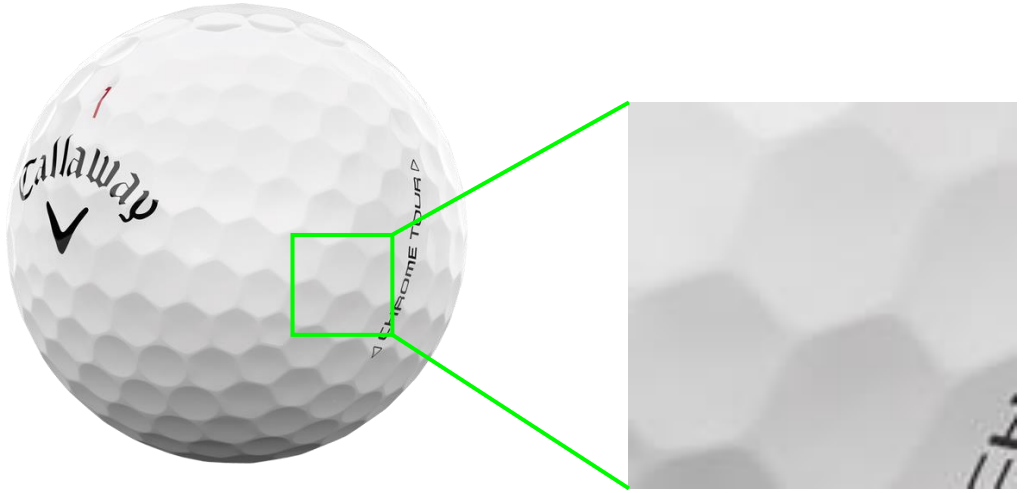
Content

- Non-Rigid Modeling
  - ➢Mass-spring system
  - ➢Position-based Dynamics
  - ➢Tetrahedral and Deformation

Mass-Spring System and Particle-based PBD Do Not Model Continuum Mechanics

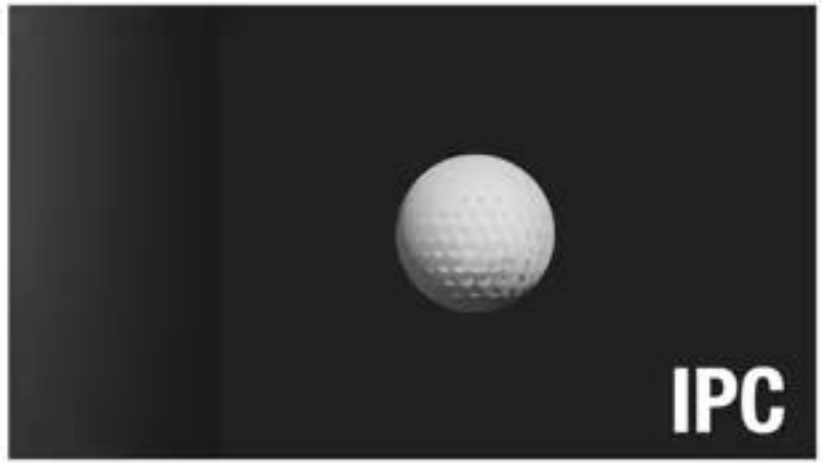


Spaces between nodes are empty: no mass and energy



Spaces are continuously filled

Common Objects are Elastic, Even Though They Look Like Rigid Bodies



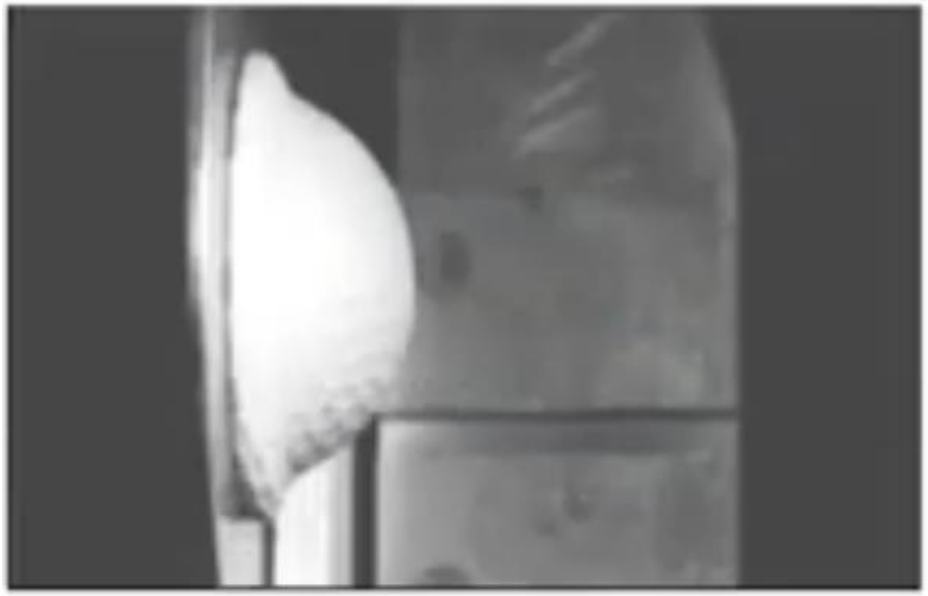
IPC Velocity Magnitude

5e-4X

Recorded Footage

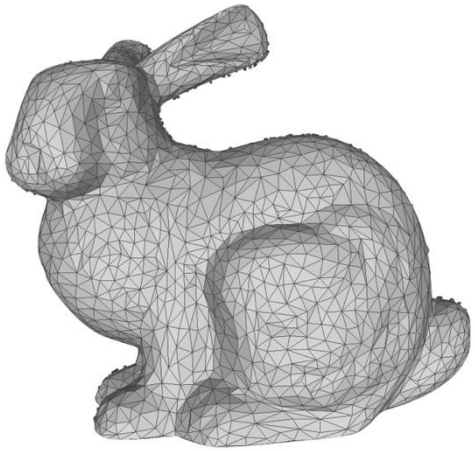
Continuum Mechanics

- How did every point in the object change shape?
- How to represent a continuum object?

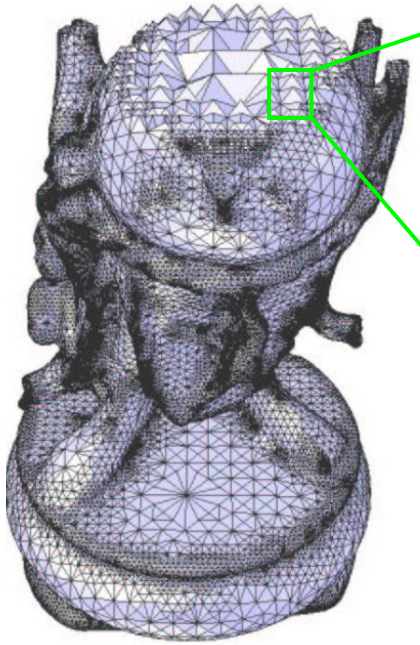


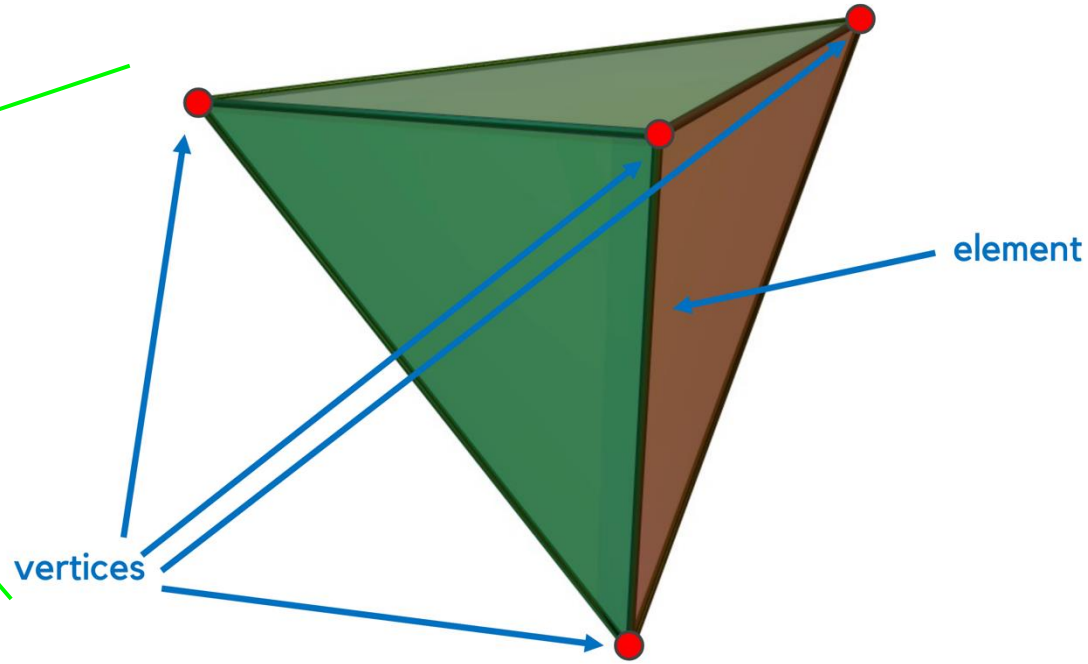
Tetrahedral Finite Elements

Mesh



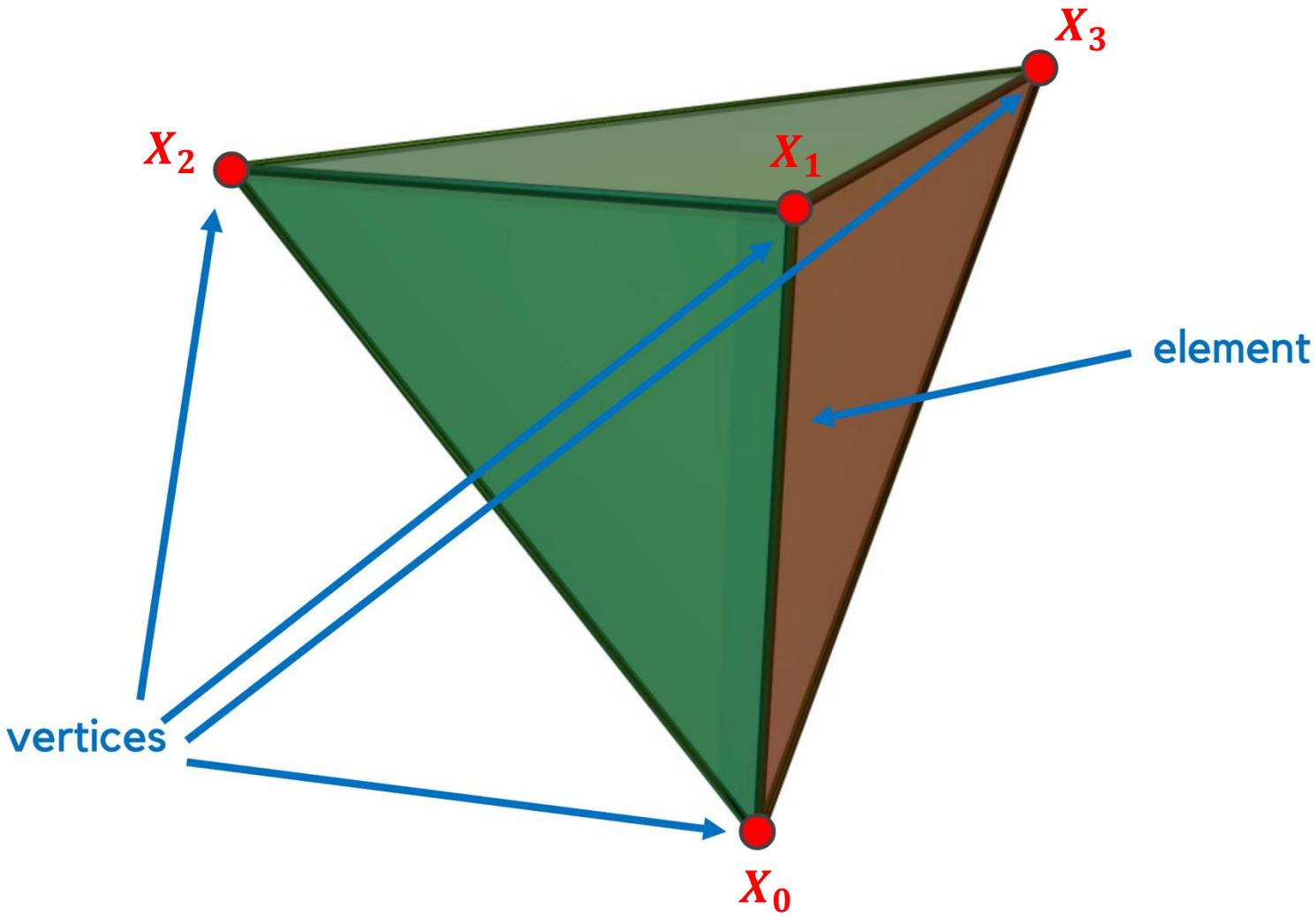
Tetrahedral Mesh





Slide adapted from D. Levin

Tetrahedral Finite Elements

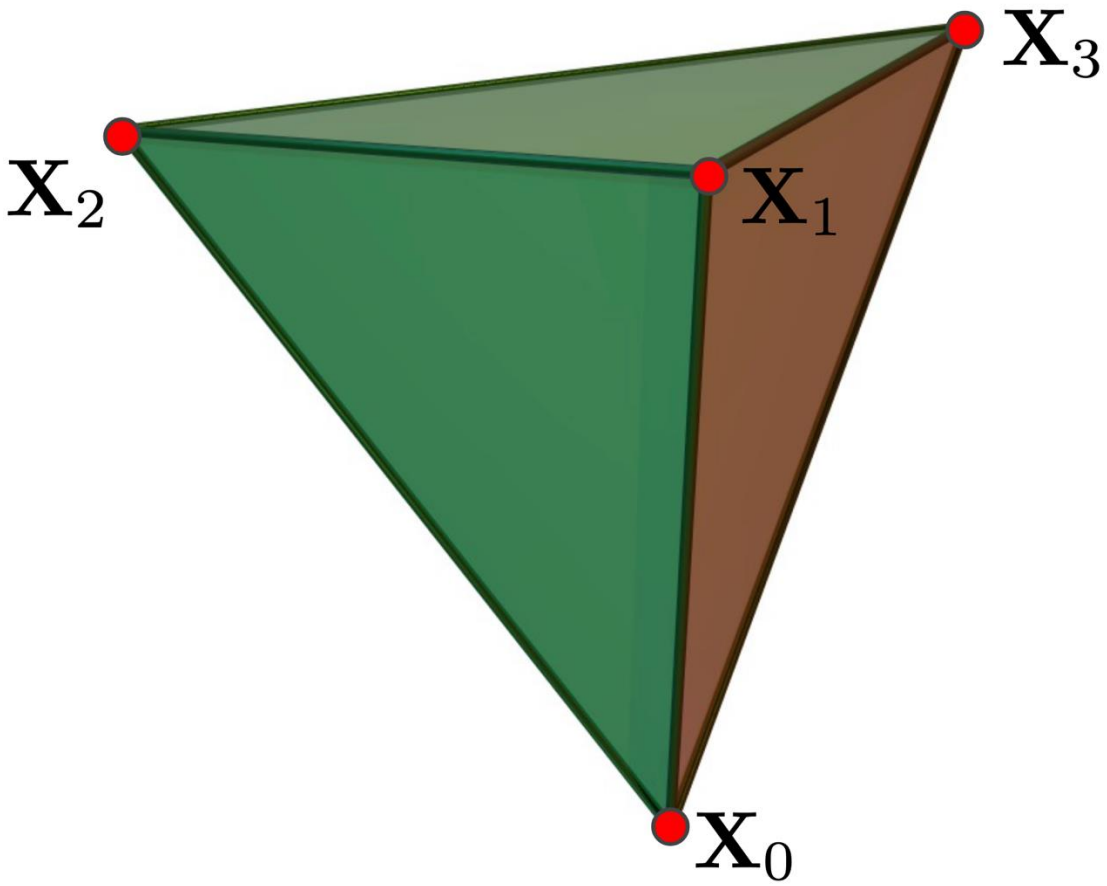


$$q = \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{pmatrix}$$

$$\dot{q} = \begin{pmatrix} \dot{X}_0 \\ \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \end{pmatrix}$$

Slide adapted from D. Levin

Generalized Coordinates for Tetrahedra Finite Elements

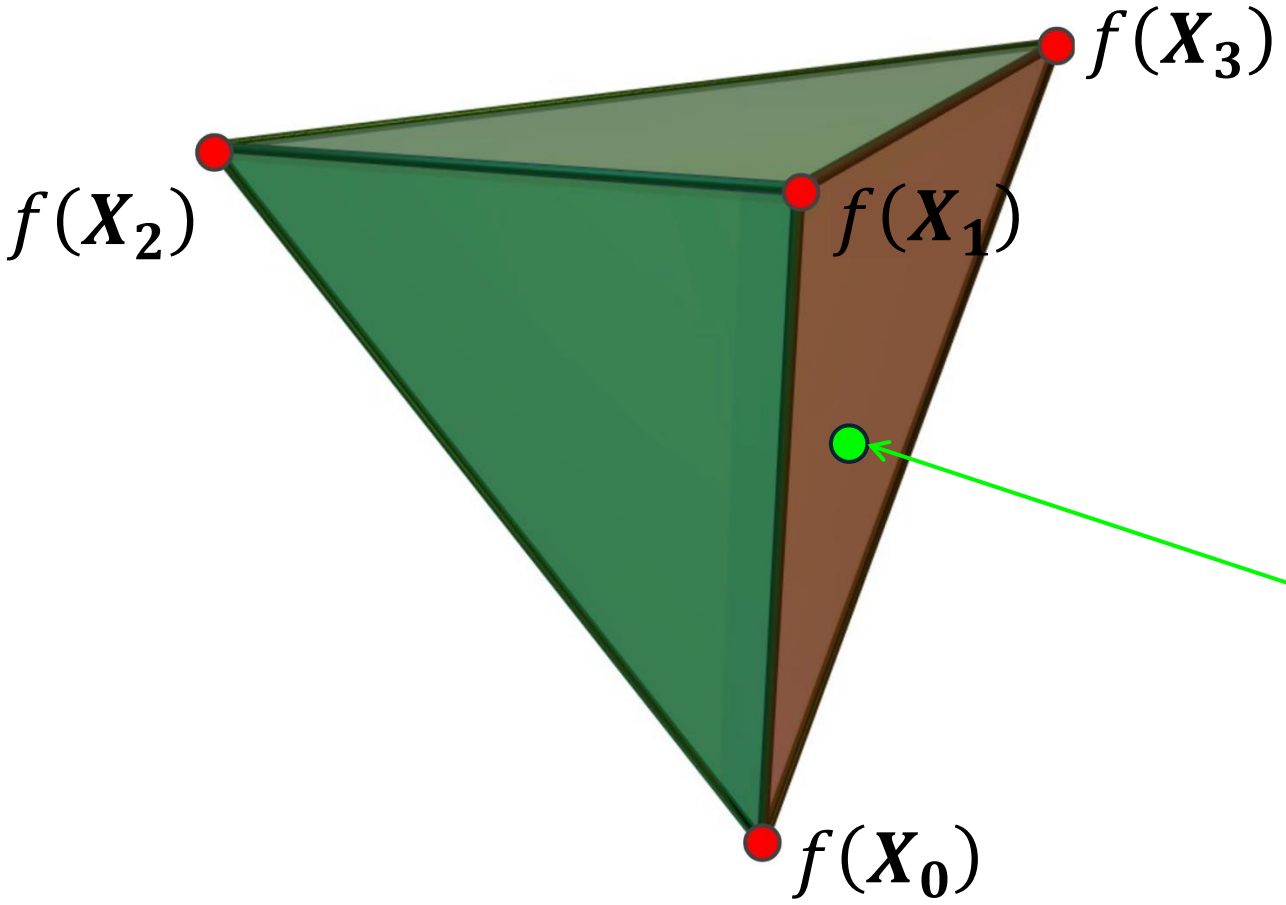


- X can be a set of parameters that specify the configuration of a physical system at each point in time (e.g. rotational angle in a pendulum system)

$$\boldsymbol{q} = \begin{pmatrix} \boldsymbol{X_0} \\ \boldsymbol{X_1} \\ \boldsymbol{X_2} \\ \boldsymbol{X_3} \end{pmatrix} \qquad \dot{\boldsymbol{q}} = \begin{pmatrix} \dot{\boldsymbol{X}}_{\boldsymbol{0}} \\ \dot{\boldsymbol{X}}_{\boldsymbol{1}} \\ \dot{\boldsymbol{X}}_{\boldsymbol{2}} \\ \dot{\boldsymbol{X}}_{\boldsymbol{3}} \end{pmatrix}$$

Slide adapted from D. Levin

Why Do We Need Such a Representation?

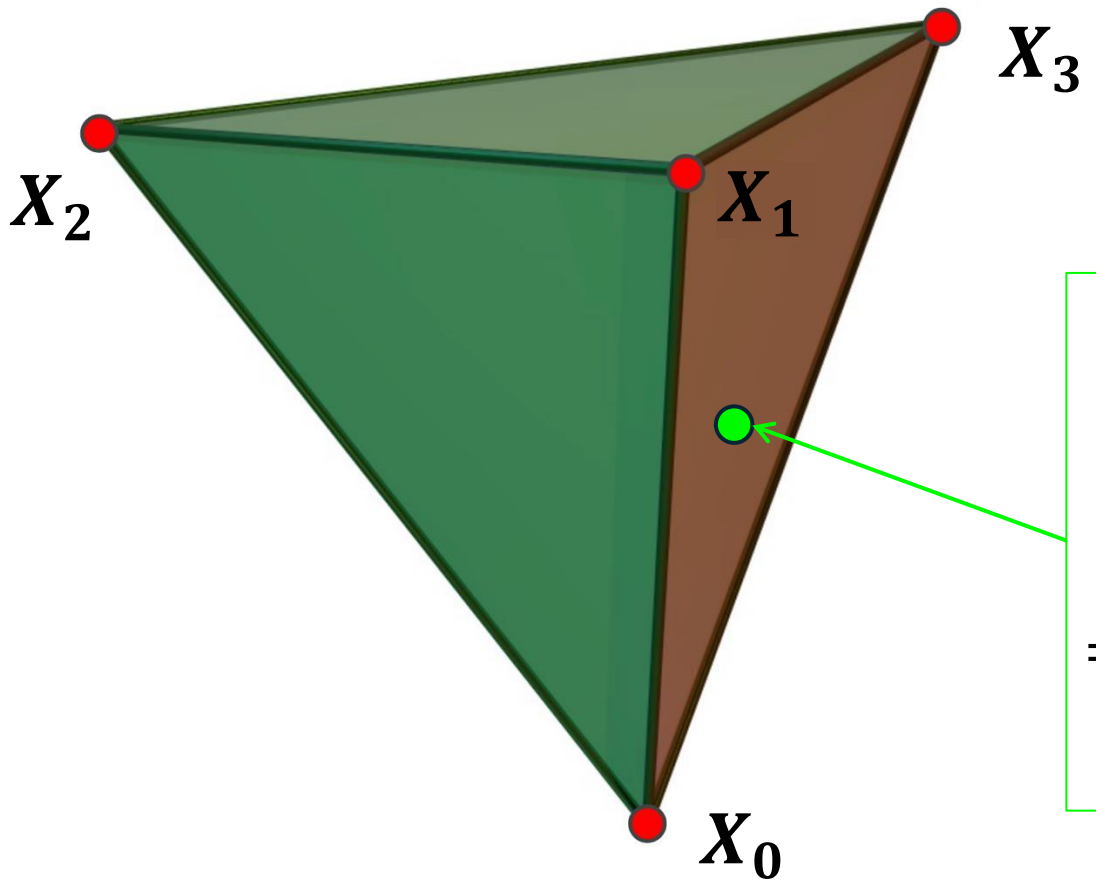


- The value of any point in a tetrahedra can be linearly interpolated
- We don't need to keep track of the deformation of every point but few vertices

$$f(X) = \sum_{i=0}^{3} f(X_i)\phi_i(X)$$

Slide adapted from D. Levin

Why Do We Need Such a Representation?



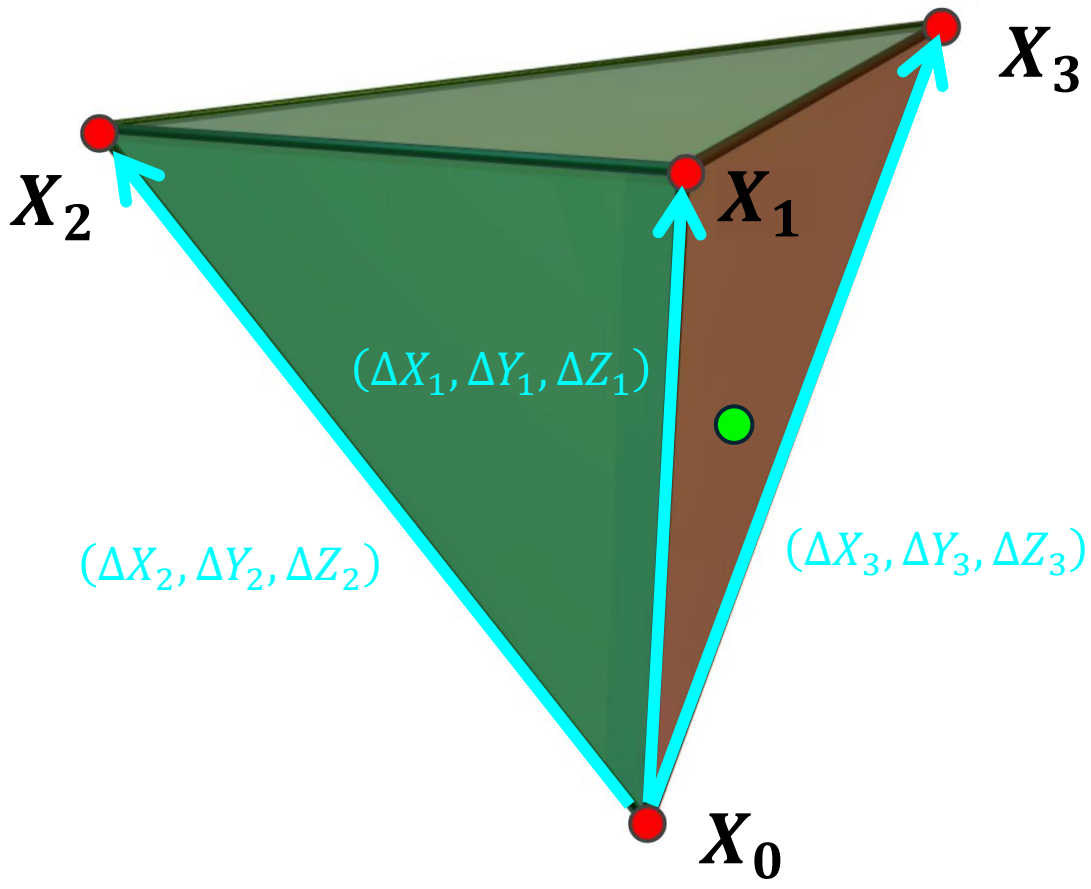
- The value of any point in a tetrahedra can be linearly interpolated

$$\boldsymbol{X} = \sum_{i=0}^{3} \boldsymbol{X_i}\phi_i(\boldsymbol{X})$$

$$\Rightarrow \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_0 & X_1 & X_2 & X_3 \\ Y_0 & Y_1 & Y_2 & Y_3 \\ Z_0 & Z_1 & Z_2 & Z_3 \end{pmatrix} \begin{pmatrix} \phi_0(\boldsymbol{X}) \\ \phi_1(\boldsymbol{X}) \\ \phi_2(\boldsymbol{X}) \\ \phi_3(\boldsymbol{X}) \end{pmatrix}$$

Slide adapted from D. Levin

Why Do We Need Such a Representation?



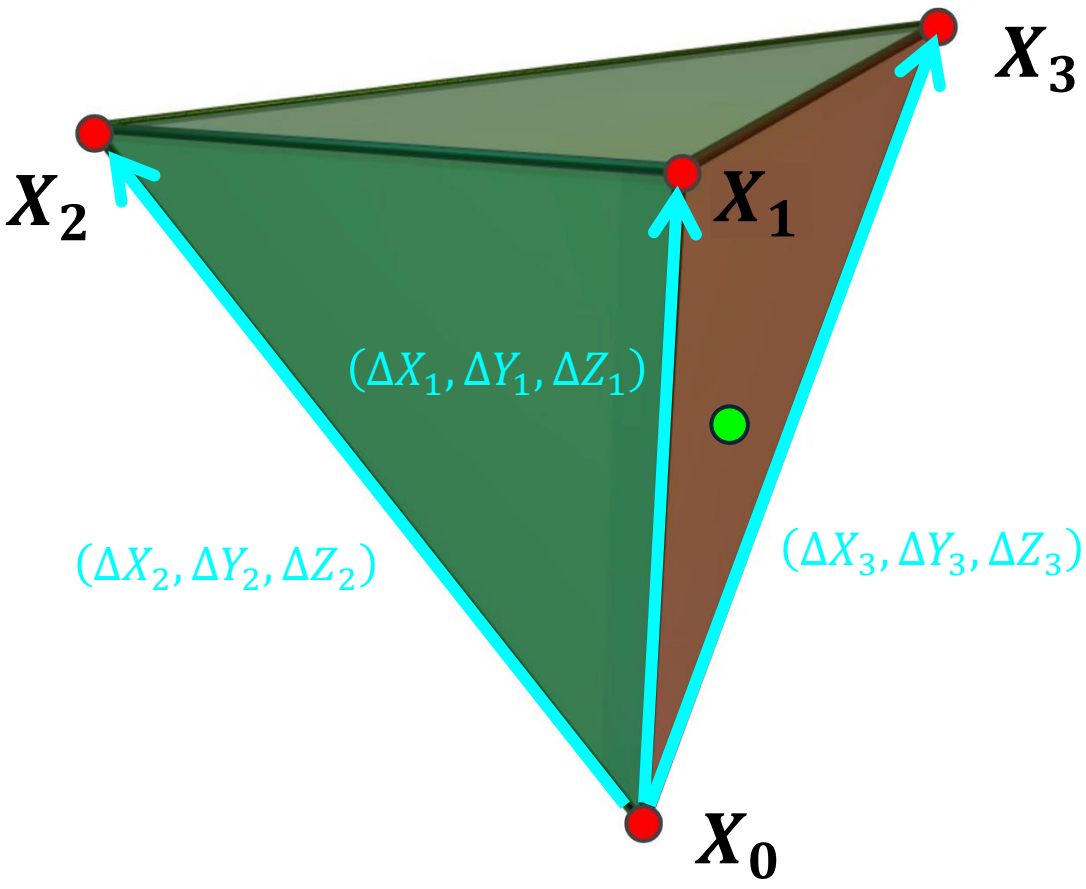
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_0 & X_1 & X_2 & X_3 \\ Y_0 & Y_1 & Y_2 & Y_3 \\ Z_0 & Z_1 & Z_2 & Z_3 \end{pmatrix} \begin{pmatrix} \phi_0(\boldsymbol{X}) \\ \phi_1(\boldsymbol{X}) \\ \phi_2(\boldsymbol{X}) \\ \phi_3(\boldsymbol{X}) \end{pmatrix}$$

Let $\phi_0(\boldsymbol{X}) = 1 - \phi_1(\boldsymbol{X}) - \phi_2(\boldsymbol{X}) - \phi_3(\boldsymbol{X})$

$$\Rightarrow \begin{pmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{pmatrix} = \begin{pmatrix} \Delta X_1 & \Delta X_2 & \Delta X_3 \\ \Delta Y_1 & \Delta Y_2 & \Delta Y_3 \\ \Delta Z_1 & \Delta Z_2 & \Delta Z_3 \end{pmatrix} \begin{pmatrix} \phi_1(\boldsymbol{X}) \\ \phi_2(\boldsymbol{X}) \\ \phi_3(\boldsymbol{X}) \end{pmatrix}$$

Slide adapted from D. Levin

Barycentric Coordinates



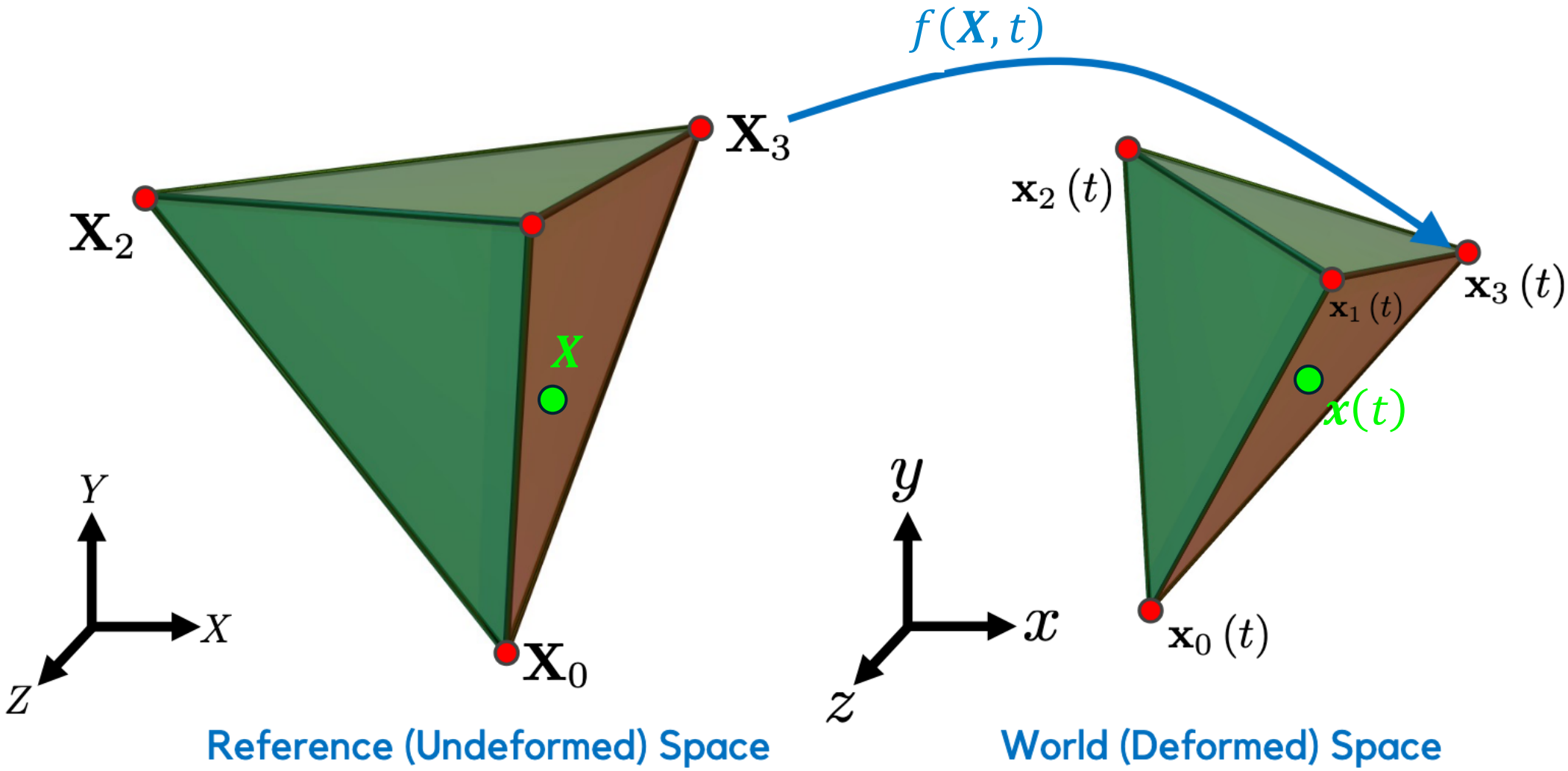
$$\begin{pmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{pmatrix} = \begin{pmatrix} \Delta X_1 & \Delta X_2 & \Delta X_3 \\ \Delta Y_1 & \Delta Y_2 & \Delta Y_3 \\ \Delta Z_1 & \Delta Z_2 & \Delta Z_3 \end{pmatrix} \begin{pmatrix} \phi_1(\boldsymbol{X}) \\ \phi_2(\boldsymbol{X}) \\ \phi_3(\boldsymbol{X}) \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} \phi_1(\boldsymbol{X}) \\ \phi_2(\boldsymbol{X}) \\ \phi_3(\boldsymbol{X}) \end{pmatrix} = \begin{pmatrix} \Delta X_1 & \Delta X_2 & \Delta X_3 \\ \Delta Y_1 & \Delta Y_2 & \Delta Y_3 \\ \Delta Z_1 & \Delta Z_2 & \Delta Z_3 \end{pmatrix}^{-1} \begin{pmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{pmatrix}$$

We can infer the interpolation weights from an undeformed tetrahedral

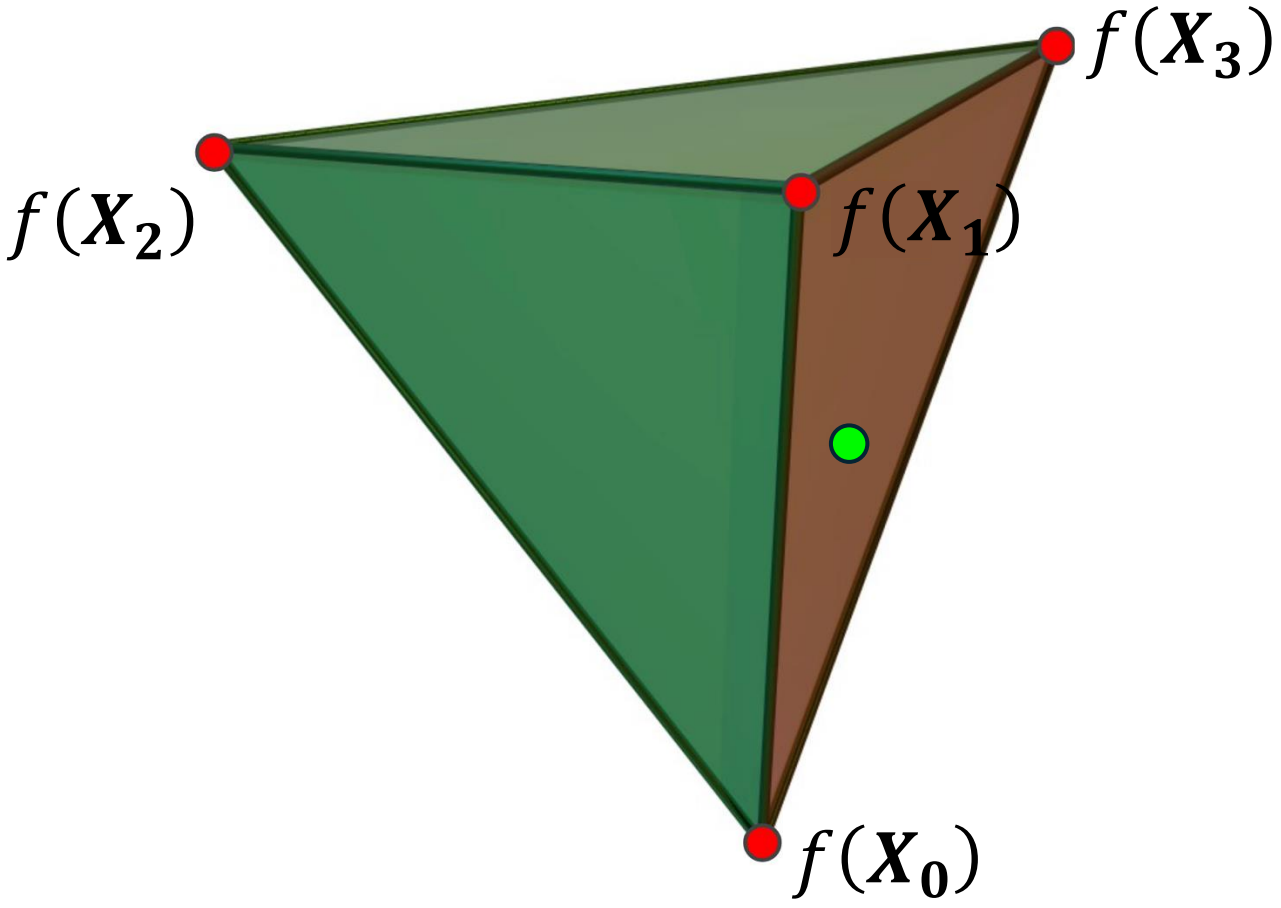
Slide adapted from D. Levin

Material Deformation



Slide adapted from D. Levin

Material Deformation



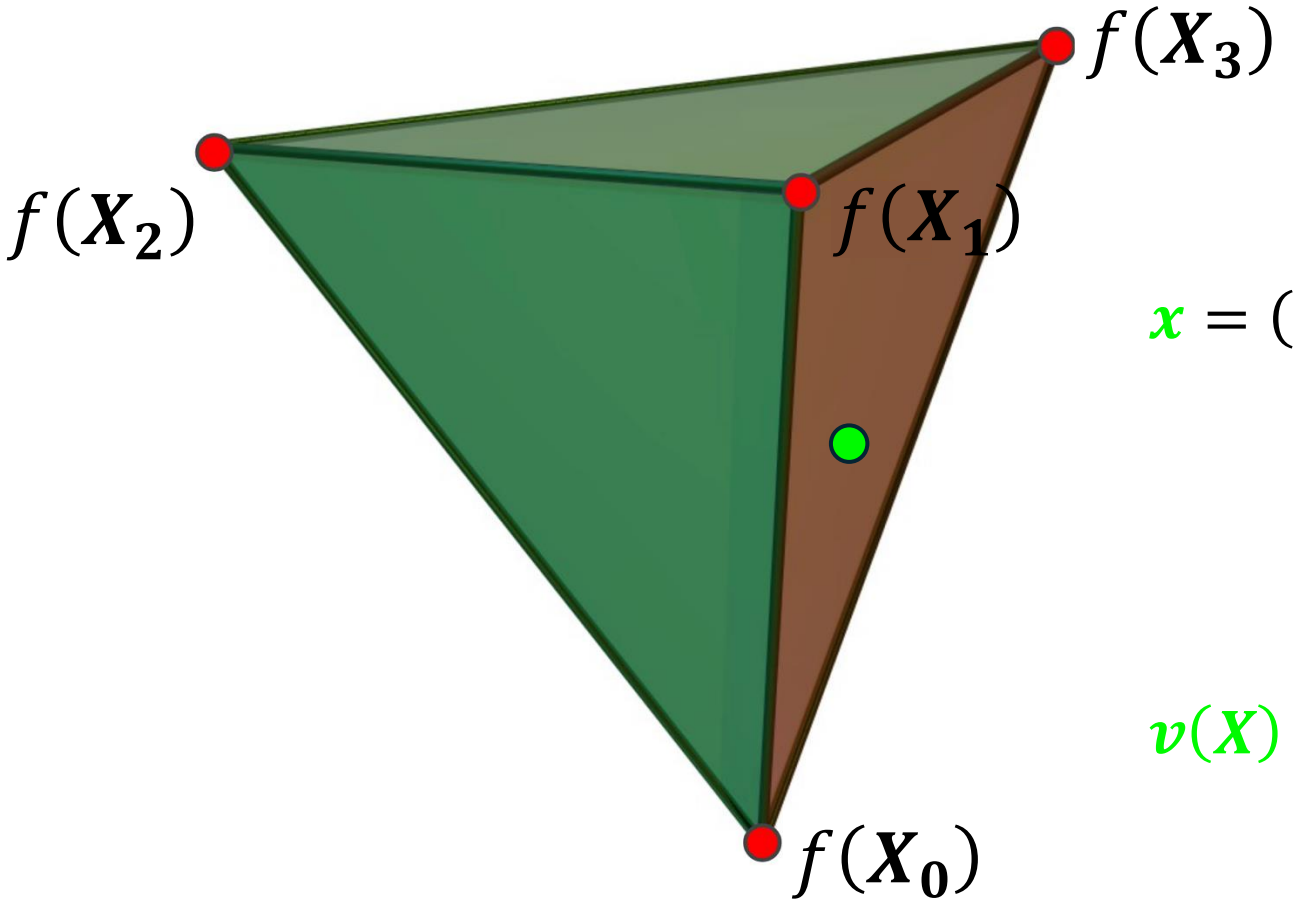
- The value of any point in a tetrahedra can be linearly interpolated
- We don't need to keep track of the deformation of every point but few vertices

$$x(t) = \sum_{i=0}^{3} x_i(t)\phi_i(X)$$

Deformed vertex position

Slide adapted from D. Levin

Material Deformation



$$x = \underbrace{(\phi_0(X) \quad \phi_1(X) \quad \phi_2(X) \quad \phi_3(X))}_{N(X)} \underbrace{\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{q}$$

$$v(X) = \dot{x} = N(X) \underbrace{\begin{pmatrix} \dot{x}_0 \\ \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix}}_{\dot{q}}$$

Slide adapted from D. Levin

In the Next Lecture, We'll Introduce a More General Framework to Derive Equations of Motion

- The Lagrangian:

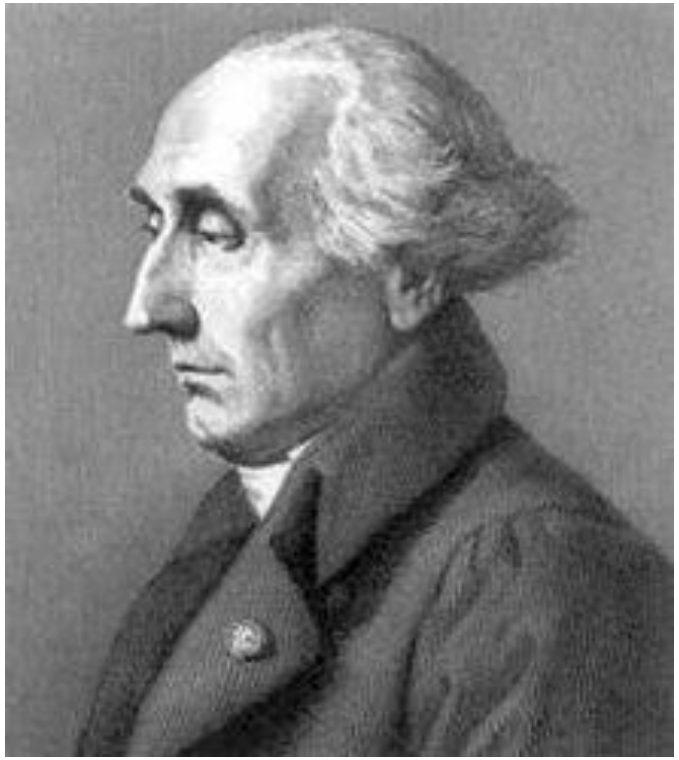
Potential energy

$$L = T - V$$

Kinetic energy

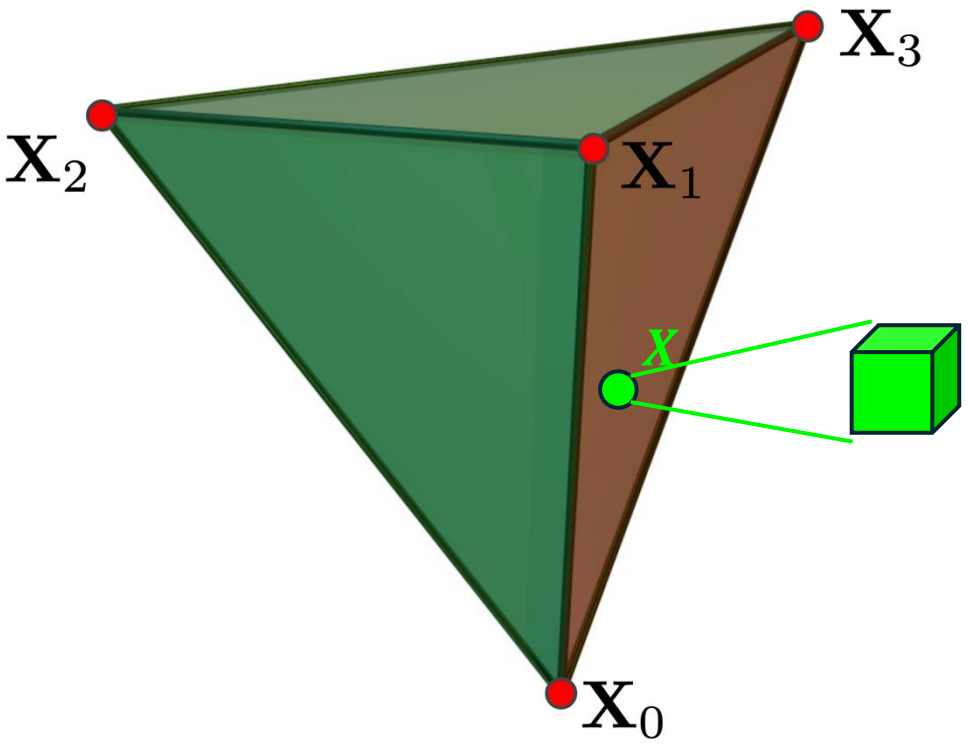
- Euler-Lagrange Equation:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{q}}} = -\frac{\partial L}{\partial \boldsymbol{q}}$$



Joseph-Louis Lagrange

Kinetic Energy of a Deformed Material



- Kinetic energy at $\boldsymbol{X}$:

$$\frac{1}{2}\rho\|v(X)\|_2^2 d\Omega$$

infinitesimal volume

unit density

- Kinetic energy of the tetrahedral

$$\int_{\Omega} \frac{1}{2}\rho\|v(X)\|_2^2 d\Omega$$

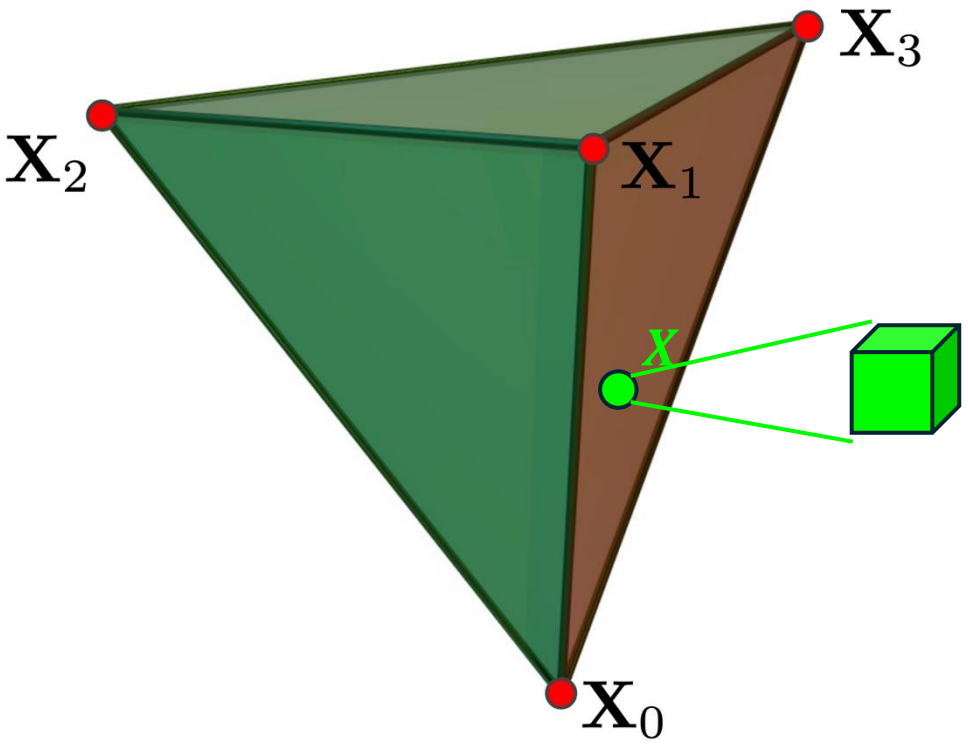
Ω → Tetralhedra domain

$$\Rightarrow \frac{1}{2}\int_{\Omega} \rho(N(\boldsymbol{X})\dot{\boldsymbol{q}})^{\mathsf{T}} N(\boldsymbol{X})\dot{\boldsymbol{q}} d\Omega$$

$$\Rightarrow \frac{1}{2}\dot{\boldsymbol{q}}^{\mathsf{T}}\left(\int_{\Omega} \rho N(\boldsymbol{X})^{\mathsf{T}} N(\boldsymbol{X}) d\Omega\right)\dot{\boldsymbol{q}}$$

Slide adapted from D. Levin

Kinetic Energy of a Deformed Material



- Kinetic energy of the tetrahedral

$$\frac{1}{2}\dot{\boldsymbol{q}}^{\mathsf{T}}\left(\int_{\Omega}\rho N(\boldsymbol{X})^{\mathsf{T}}N(\boldsymbol{X})d\Omega\right)\dot{\boldsymbol{q}}$$

$$\int_{\Omega}\rho\begin{pmatrix}\phi_0\phi_0 I & \phi_0\phi_1 I & \phi_0\phi_2 I & \phi_0\phi_3 I\\ \phi_1\phi_0 I & \phi_1\phi_1 I & \phi_1\phi_2 I & \phi_1\phi_3 I\\ \phi_2\phi_0 I & \phi_2\phi_1 I & \phi_2\phi_2 I & \phi_2\phi_3 I\\ \phi_3\phi_0 I & \phi_3\phi_1 I & \phi_3\phi_2 I & \phi_3\phi_3 I\end{pmatrix}d\Omega$$

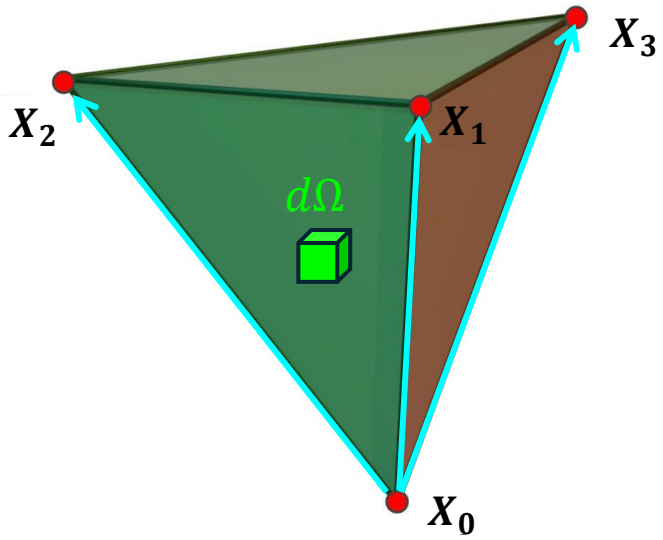
- Evaluate at each term separately:

$$\int_{\Omega}\rho\phi_r\phi_s I d\Omega$$

Slide adapted from D. Levin

The Differential Volume Element

- Integral over the tetrahedral domain Ω: $\int_{\Omega} \rho \phi_r \phi_s I d\Omega$
- The integral is difficult to calculate using the Cartesian coordinates
- Idea: Transform the Cartesian coordinates into Barycentric coordinates



We have $\phi_0(\boldsymbol{X}) = 1 - \phi_1(\boldsymbol{X}) - \phi_2(\boldsymbol{X}) - \phi_3(\boldsymbol{X})$

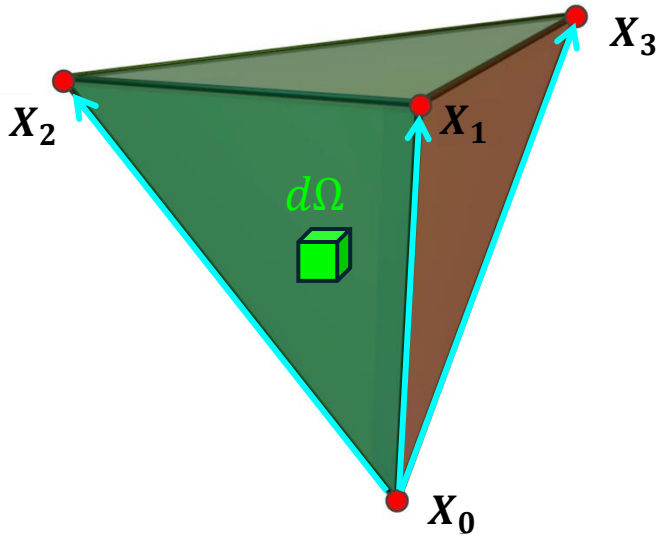
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_0 & X_1 & X_2 & X_3 \\ Y_0 & Y_1 & Y_2 & Y_3 \\ Z_0 & Z_1 & Z_2 & Z_3 \end{pmatrix} \begin{pmatrix} \phi_0(\boldsymbol{X}) \\ \phi_1(\boldsymbol{X}) \\ \phi_2(\boldsymbol{X}) \\ \phi_3(\boldsymbol{X}) \end{pmatrix}$$

Cartesian coordinates

Barycentric coordinates

The Differential Volume Element

- Integral over the tetrahedral domain Ω: $\int_{\Omega} \rho \phi_r \phi_s I d\Omega$
- The integral is difficult to calculate using the Cartesian coordinates
- Idea: Transform the Cartesian coordinates into Barycentric coordinates



$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_0 & (X_1 - X_0)\phi_1 & (X_2 - X_0)\phi_2 & (X_3 - X_0)\phi_3 \\ Y_0 & (Y_1 - Y_0)\phi_1 & (Y_2 - Y_0)\phi_2 & (Y_3 - Y_0)\phi_3 \\ Z_0 & (Z_1 - Z_0)\phi_1 & (Z_2 - Z_0)\phi_2 & (Z_3 - Z_0)\phi_3 \end{pmatrix}$$

Remember that We Transformed Two Coordinate System with Jacobian $\boldsymbol{J_{u_k}}$

- Projection is not affine mapping

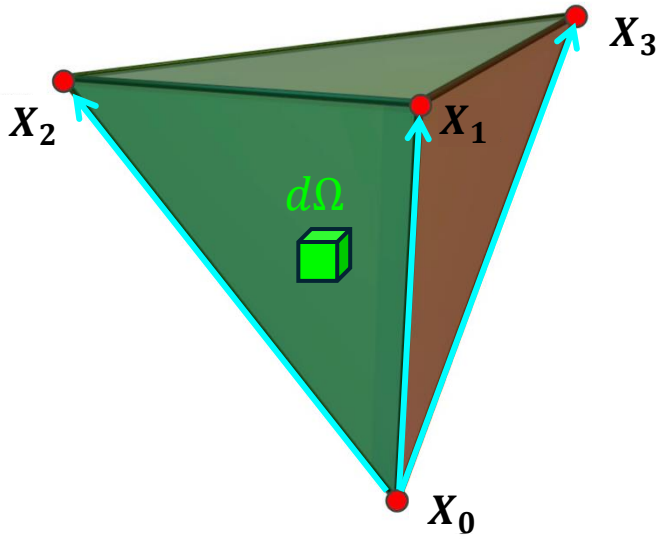
$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = m(\boldsymbol{u}) = \begin{pmatrix} u_0/u_2 \\ u_1/u_2 \\ \|\boldsymbol{u}\| \end{pmatrix}$$

- We can approximate with 1st order Taylor Expansion at mean $\boldsymbol{u_k}$ and $\boldsymbol{x_k}$ of two spaces as:

$$m_{\boldsymbol{u_k}}(\boldsymbol{u}) = \boldsymbol{x_k} + J_{\boldsymbol{u_k}} \cdot (\boldsymbol{u} - \boldsymbol{u_k}), \qquad \text{where } J_{\boldsymbol{u_k}} = \frac{\partial m}{\partial \boldsymbol{u}}(\boldsymbol{u_k})$$

The Differential Volume Element

- Integral over the tetrahedral domain Ω: $\int_{\Omega} \rho \phi_r \phi_s I d\Omega$
- The integral is difficult to calculate using the Cartesian coordinates
- Idea: Transform the Cartesian coordinates into Barycentric coordinates



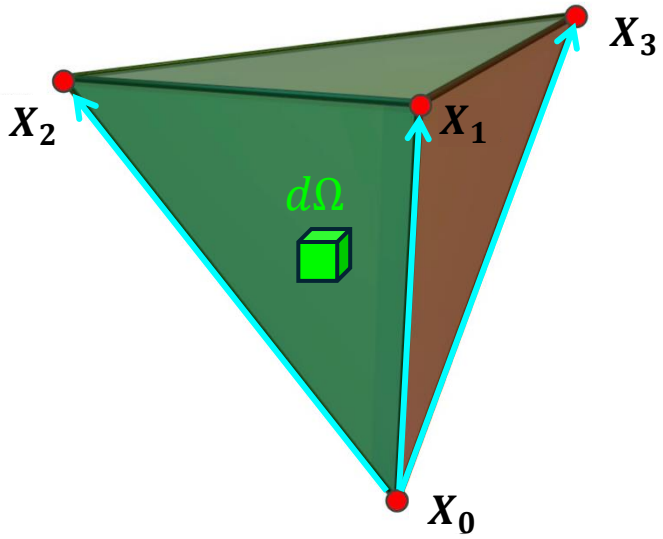
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_0 & (X_1 - X_0)\phi_1 & (X_2 - X_0)\phi_2 & (X_3 - X_0)\phi_3 \\ Y_0 & (Y_1 - Y_0)\phi_1 & (Y_2 - Y_0)\phi_2 & (Y_3 - Y_0)\phi_3 \\ Z_0 & (Z_1 - Z_0)\phi_1 & (Z_2 - Z_0)\phi_2 & (Z_3 - Z_0)\phi_3 \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + \boldsymbol{J} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{pmatrix}$$

$\phi_1 = \phi_2 = \phi_3 = 0$ evaluated at $\boldsymbol{X_0}$

The Differential Volume Element

- Integral over the tetrahedral domain Ω: $\int_{\Omega} \rho \phi_r \phi_s I d\Omega$
- The integral is difficult to calculate using the Cartesian coordinates
- Idea: Transform the Cartesian coordinates into Barycentric coordinates



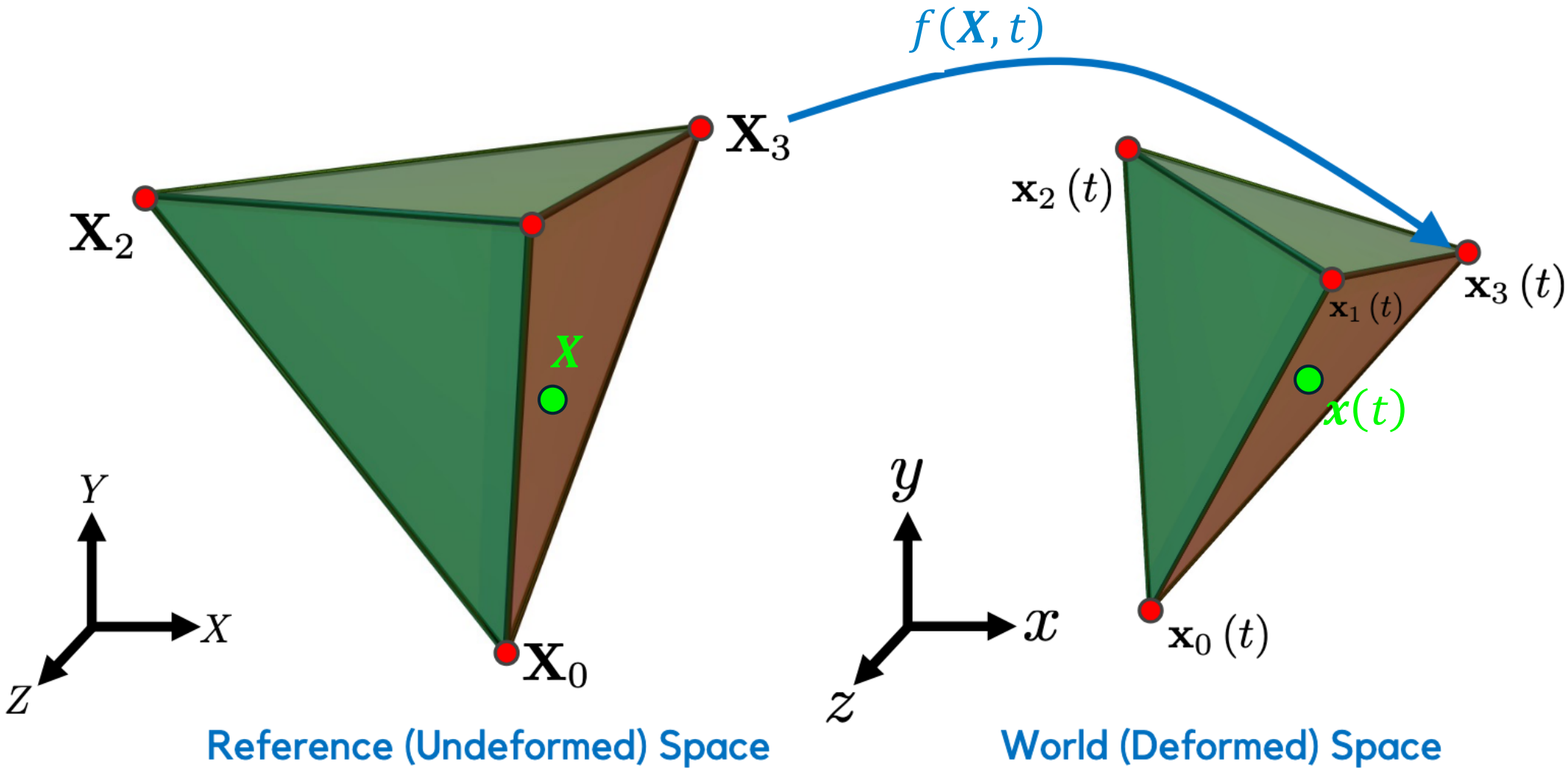
- Let V denotes the tetrahedron's volume, we have:

$$dV = |\det \boldsymbol{J}| d\phi_1 d\phi_2 d\phi_3 = 6V d\phi_1 d\phi_2 d\phi_3$$

- We can calculate the integral:

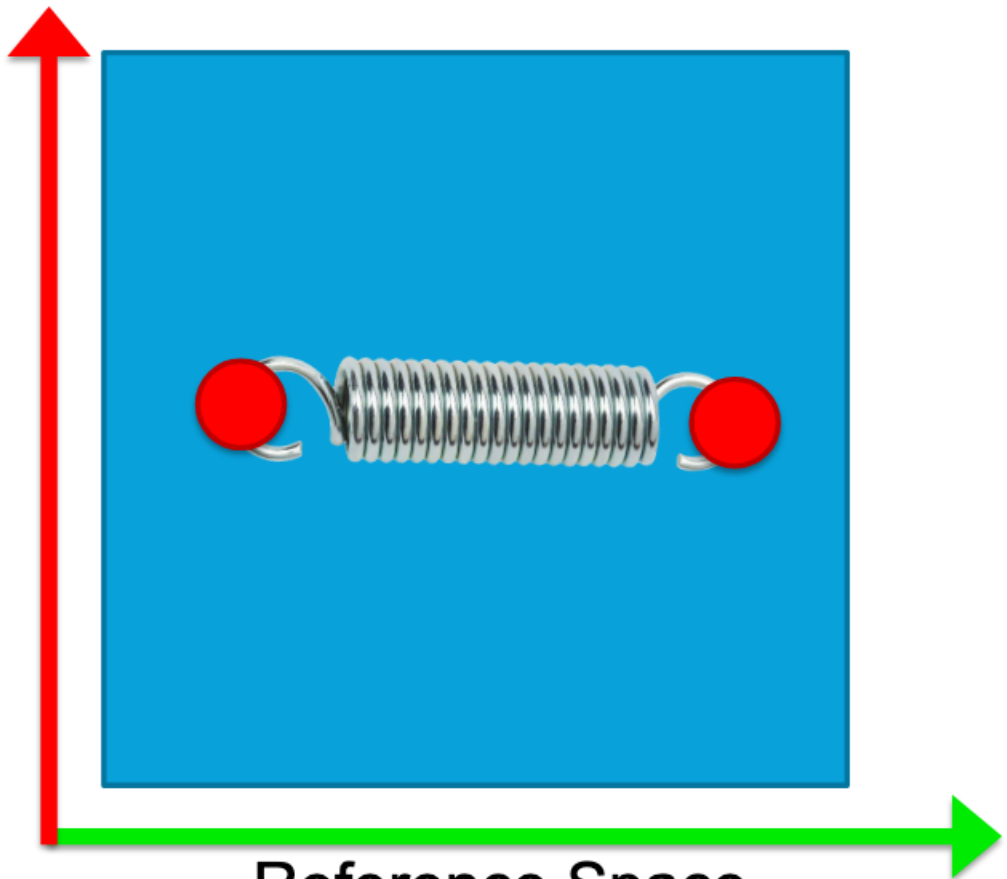
$$\int_{\Omega} \rho \phi_r \phi_s I d\Omega = 6\rho V \int_0^1 \int_0^{1-\phi_1} \int_0^{1-\phi_1-\phi_2} (\phi_r \phi_s) d\phi_1 d\phi_2 d\phi_3$$

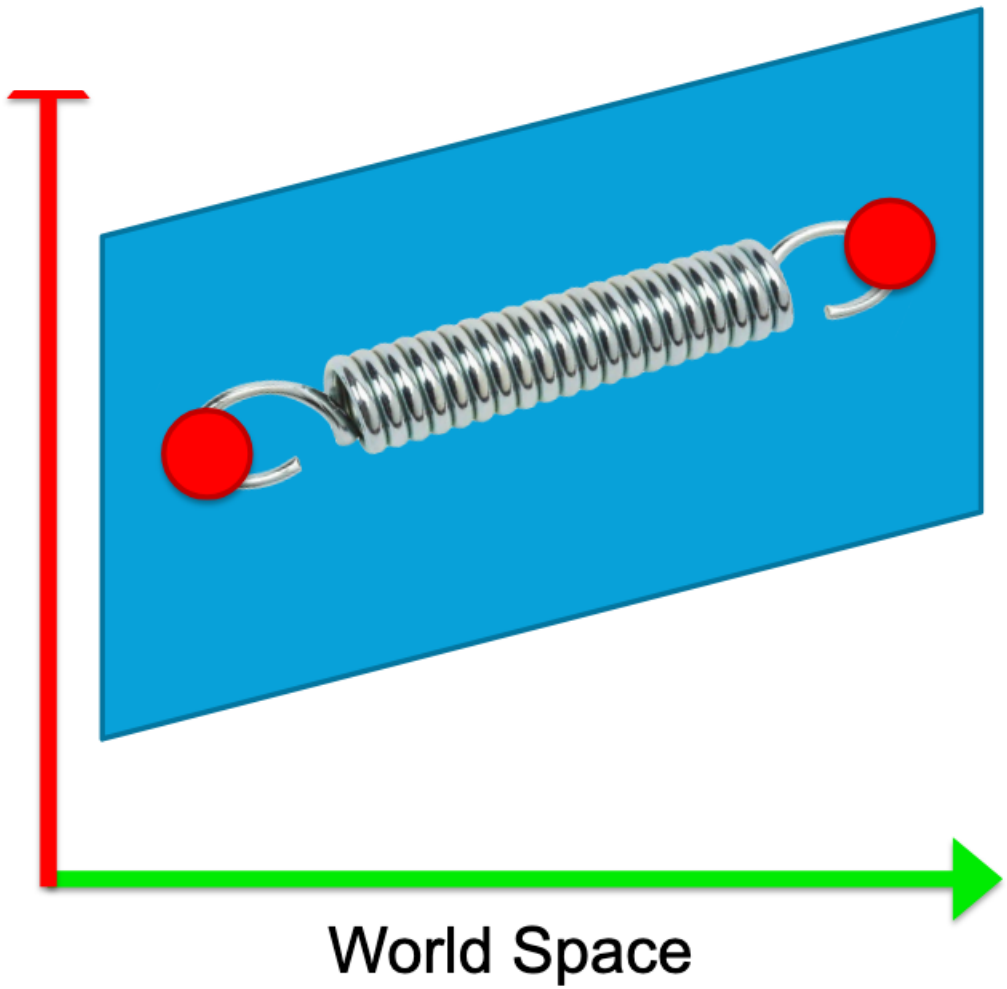
Potential Energy of Material Deformation



Slide adapted from D. Levin

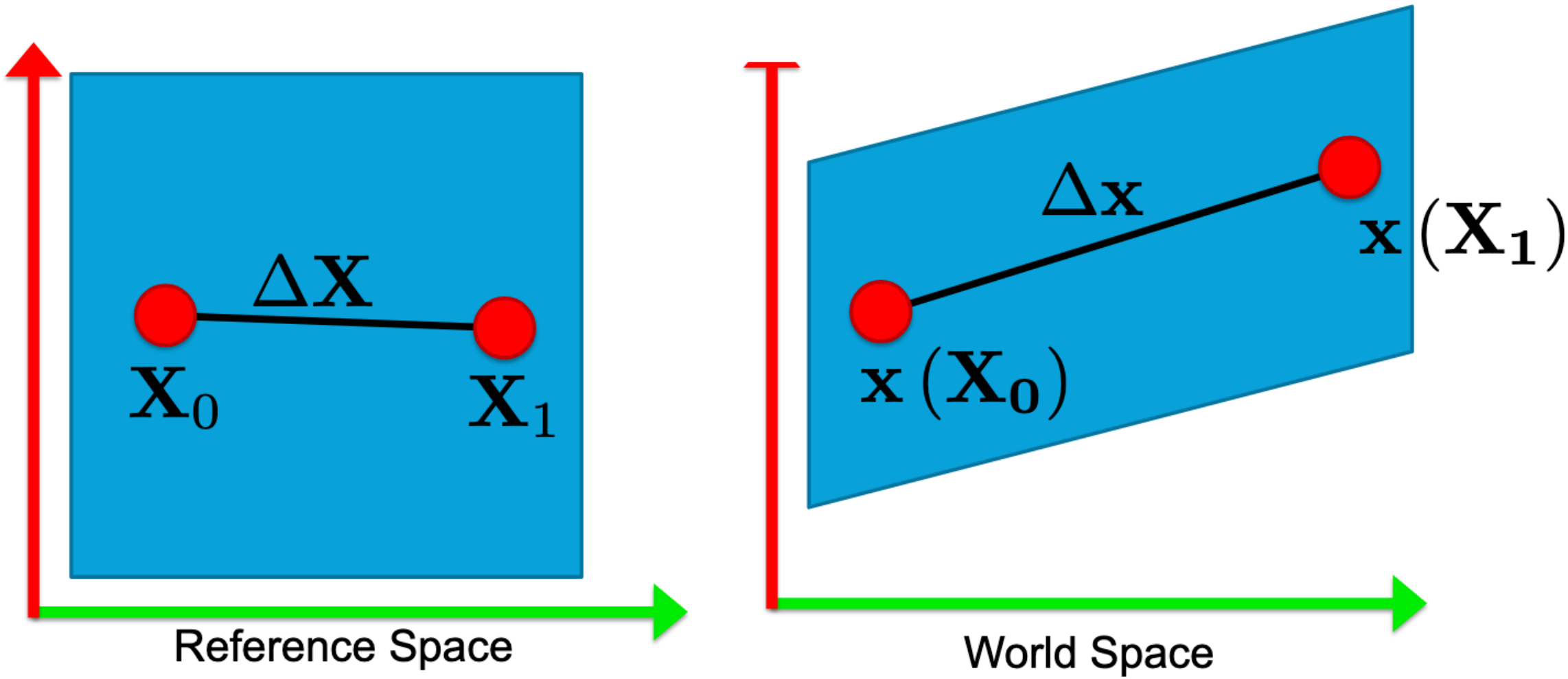
A Closer Look at Deformation





Slide adapted from D. Levin

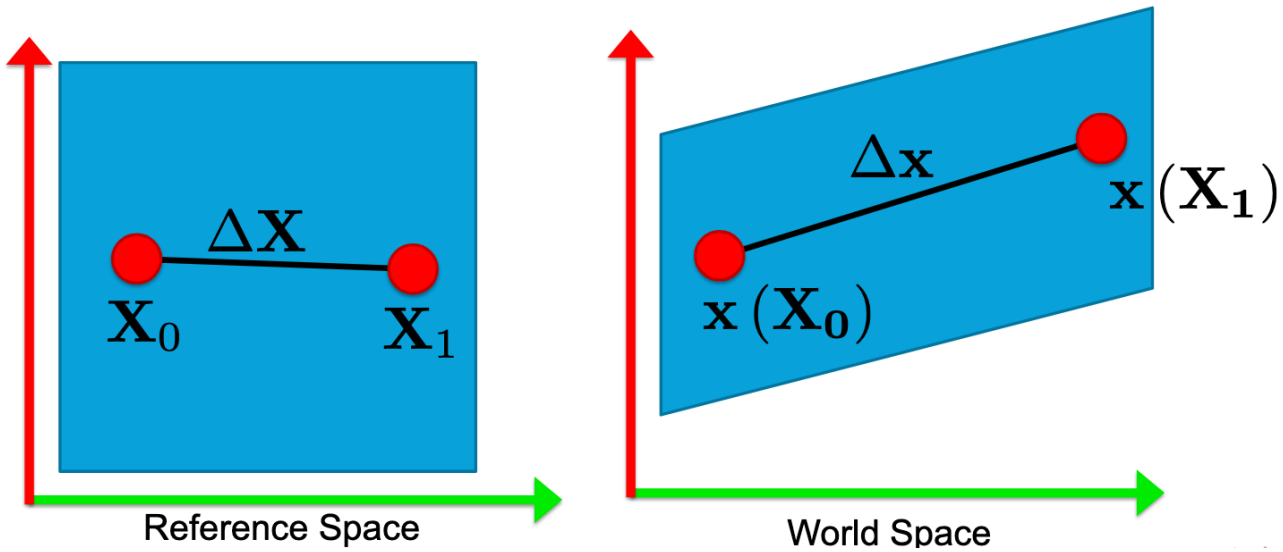
A Closer Look at Deformation



$$\Delta \mathbf{x} = \mathbf{x}(\mathbf{X}_1) - \mathbf{x}(\mathbf{X}_0)$$

Slide adapted from D. Levin

A Closer Look at Deformation



$$\Delta \boldsymbol{x} = f(\boldsymbol{X}_1) - f(\boldsymbol{X}_0)$$
$$\Rightarrow \Delta \boldsymbol{x} = f(\boldsymbol{X}_0 + \Delta \boldsymbol{X}) - f(\boldsymbol{X}_0)$$
$$\Rightarrow \Delta \boldsymbol{x} \approx f(\boldsymbol{X}_0) + \frac{\partial f}{\partial \boldsymbol{X}} \Delta \boldsymbol{X} - f(\boldsymbol{X}_0)$$

Deformation gradient $\boldsymbol{F}$

- Deformed length squared:
$$l^2 = \Delta \boldsymbol{x}^\mathsf{T} \Delta \boldsymbol{x}$$
- Rest length squared:
$$l_0^2 = \Delta \boldsymbol{X}^\mathsf{T} \Delta \boldsymbol{X}$$
- Strain:
$$l^2 - l_0^2 = \Delta \boldsymbol{x}^\mathsf{T} \Delta \boldsymbol{x} - \Delta \boldsymbol{X}^\mathsf{T} \Delta \boldsymbol{X}$$
$$= \Delta \boldsymbol{X}^\mathsf{T} \boldsymbol{F}^\mathsf{T} \boldsymbol{F} \Delta \boldsymbol{X} - \Delta \boldsymbol{X}^\mathsf{T} \Delta \boldsymbol{X}$$
$$= \Delta \boldsymbol{X}^\mathsf{T} (\boldsymbol{F}^\mathsf{T} \boldsymbol{F} - I) \Delta \boldsymbol{X}$$
- Unit potential energy resulted from strain: $\Psi\big(\boldsymbol{F}(\boldsymbol{X})\big)$

Slide adapted from D. Levin

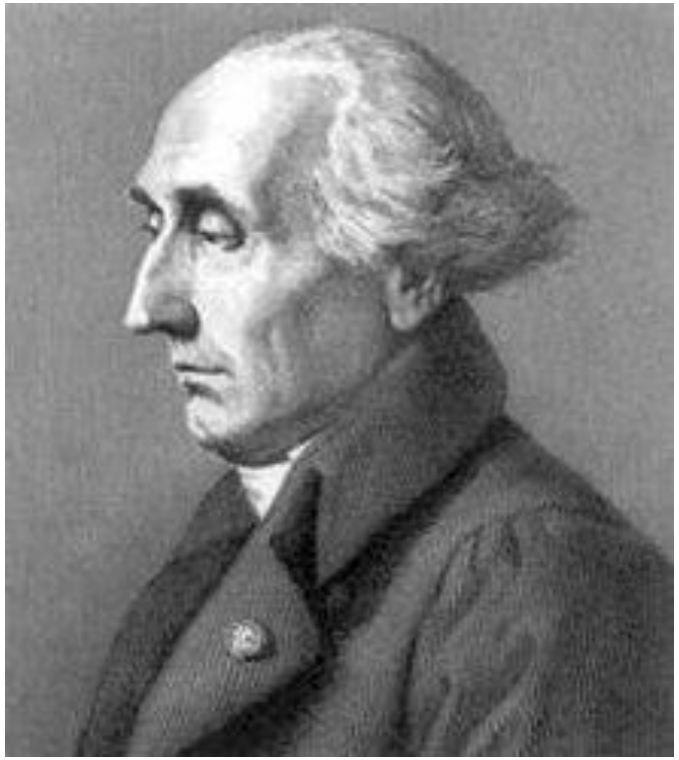
In the Next Lecture, We'll Introduce a More General Framework to Derive Equations of Motion

- The Lagrangian:

$$L = T - V$$

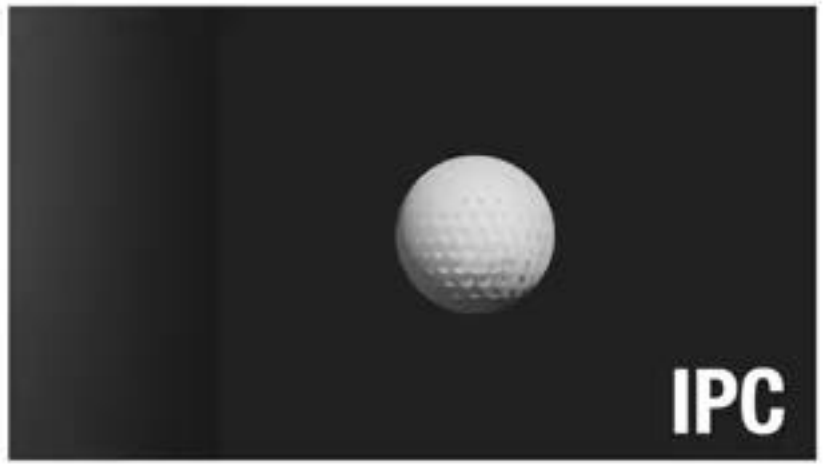
$$T = \frac{1}{2}\dot{\boldsymbol{q}}^{\mathsf{T}}\left(\int_{\Omega} \rho N(\boldsymbol{X})^{\mathsf{T}} N(\boldsymbol{X}) d\Omega\right)\dot{\boldsymbol{q}}$$

$$V = \sum_{j=0}^{m-1} vol_j \cdot \Psi\left(F_j(q_j)\right)$$



Joseph-Louis Lagrange

Why Do We Need Continuum Mechanics?



IPC Velocity Magnitude

5e-4X

Recorded Footage

What We Will Cover the Next Week

- Physical modeling
  - Lagrangian Mechanics
  - Contact Modeling
  - Material Point Methods