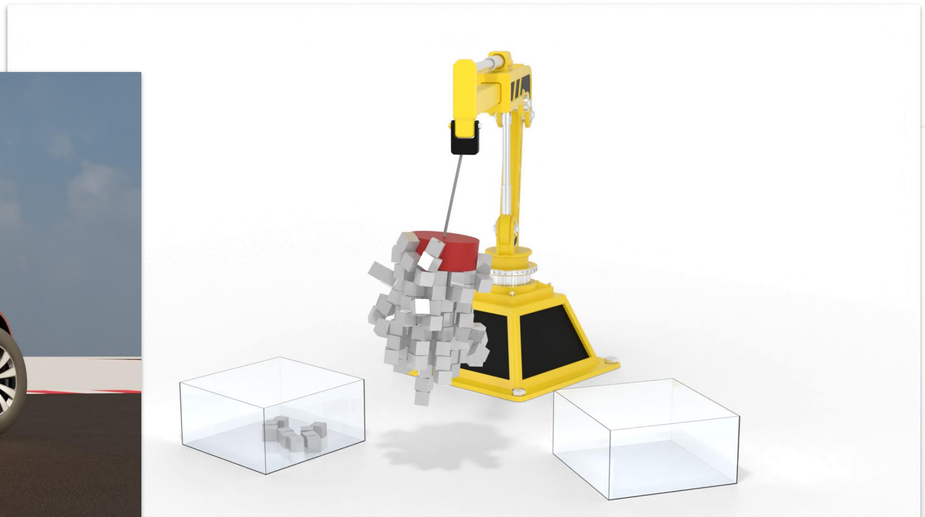
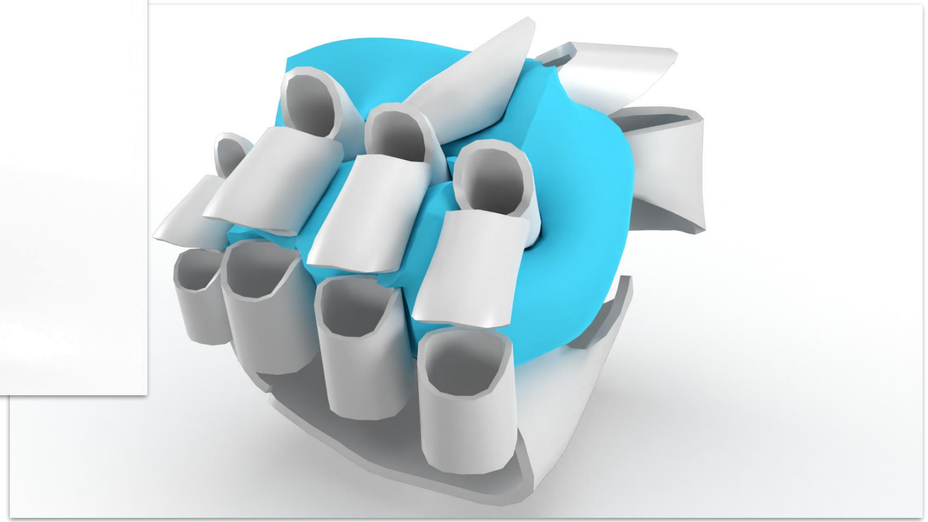
Part I: Energy-Based Simulation Foundations





Fabian Löschner [1]
Stefan Rhys Jeske [1]
José Antonio Fernández-Fernández [1]
Jan Bender [1]

[1] RWTH AACHEN UNIVERSITY

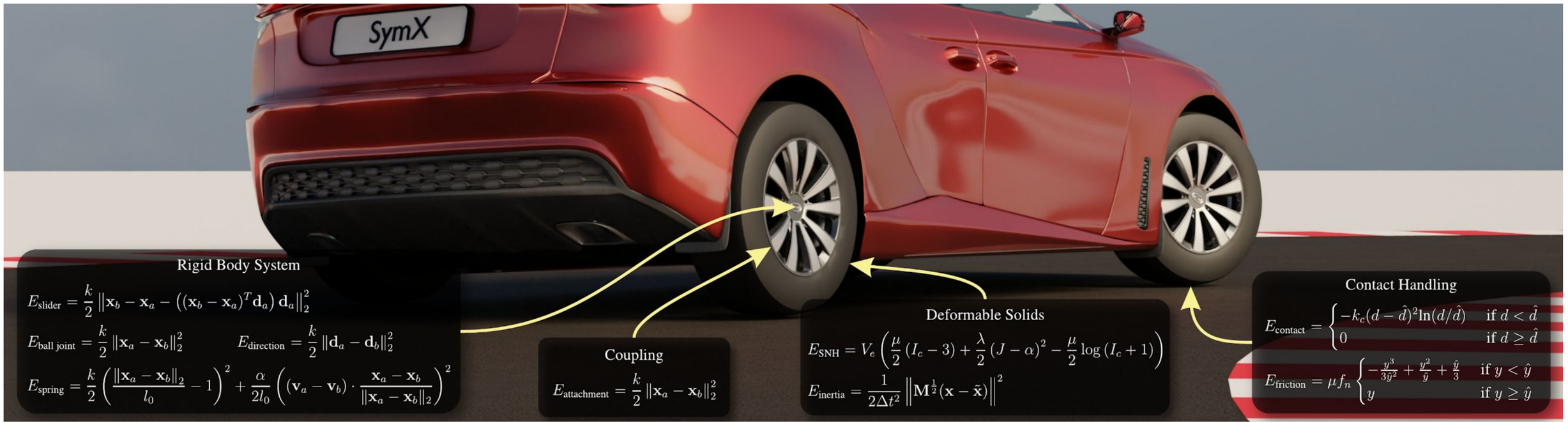
EUROGRAPHICS AACHEN 2026

Motivation

Why Energy-Based?

1. Unified formulation in terms of scalar energy potentials
   - Strongly coupled, fully implicit
2. Optimization > Root finding
   - Ensures progress and validity

1. Unified Scalar Formulation

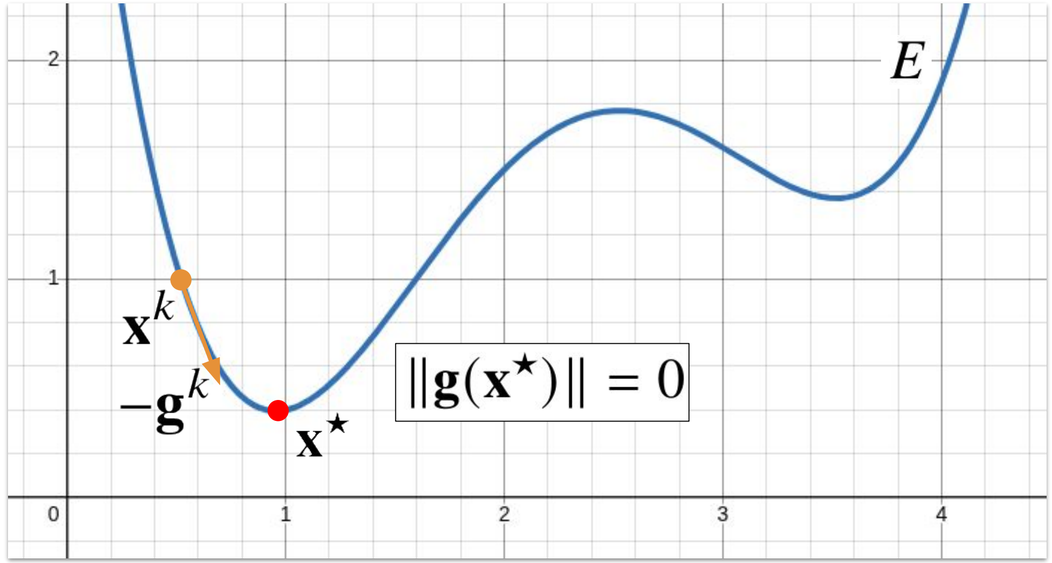


SymX [Fernández-Fernández et al. 2025]

$$E_{\text{Total}} = E_{\text{Inertia}} + E_{\text{slider}} + E_{\text{ball}} + E_{\text{spring}} + E_{\text{dir}} + E_{\text{attach}} + E_{\text{SNH}} + E_{\text{Contact}} + E_{\text{Friction}}$$

2. Robust Optimization

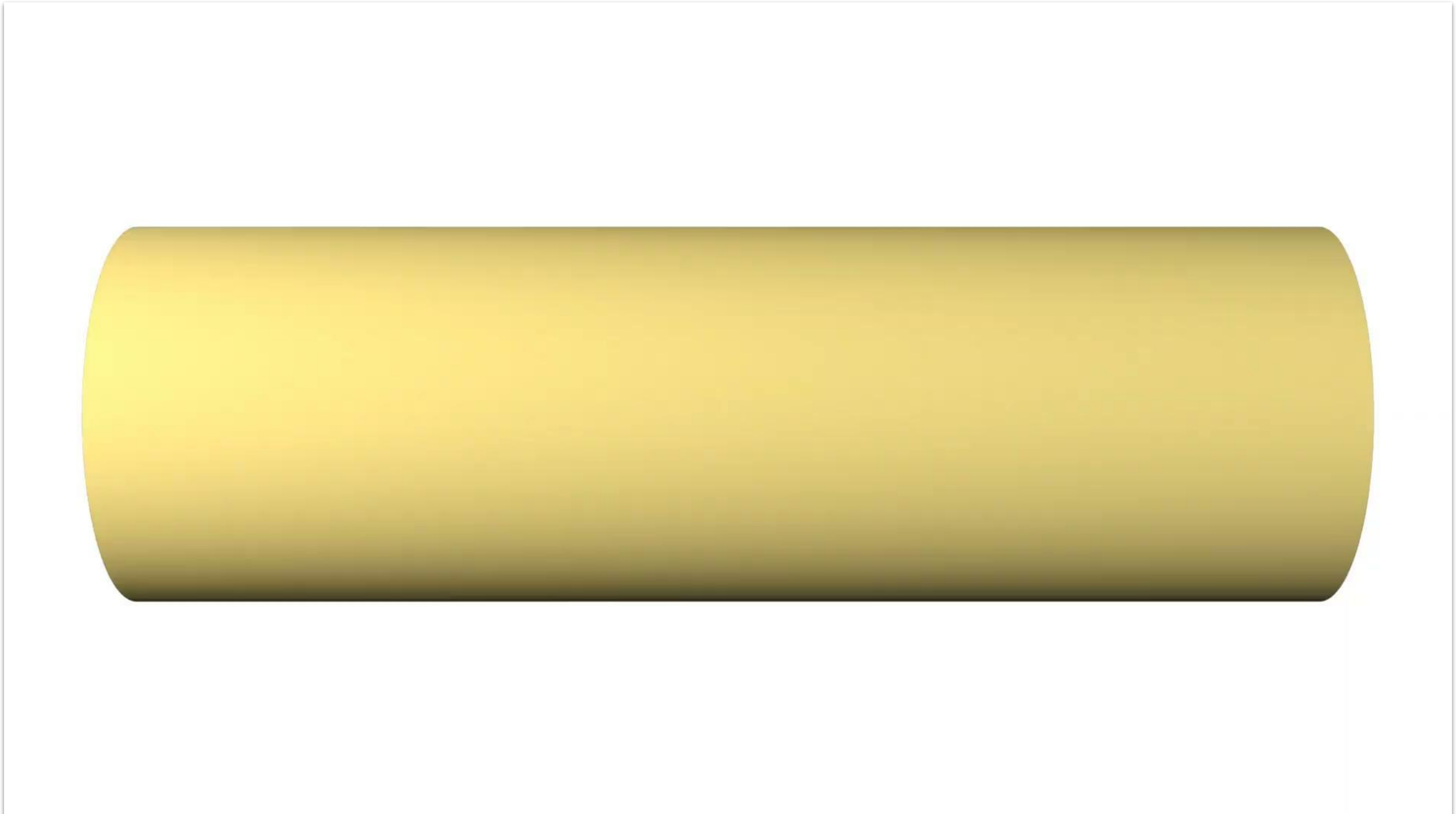
- Balance of forces at **grad(E) = 0**
- E is sufficiently continuous and differentiable
- *“Slide down to the solution”*



Numerical infrastructure:

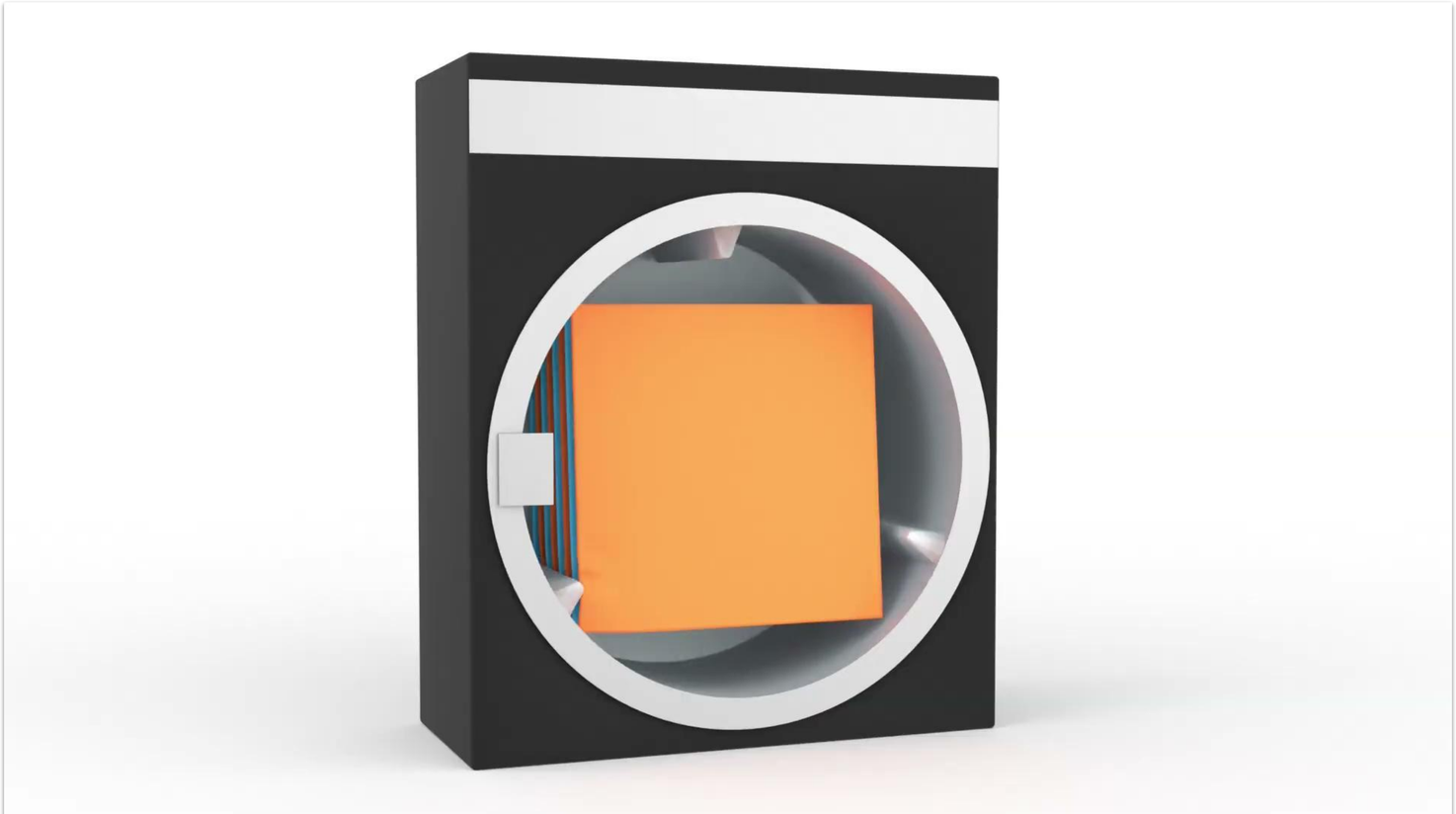
- Ensure descent directions
- Ensure admissibility
- Ensure sufficient descent

Examples



Progressively Projected Newton’s Method [Fernández-Fernández et al. 2026]

Examples

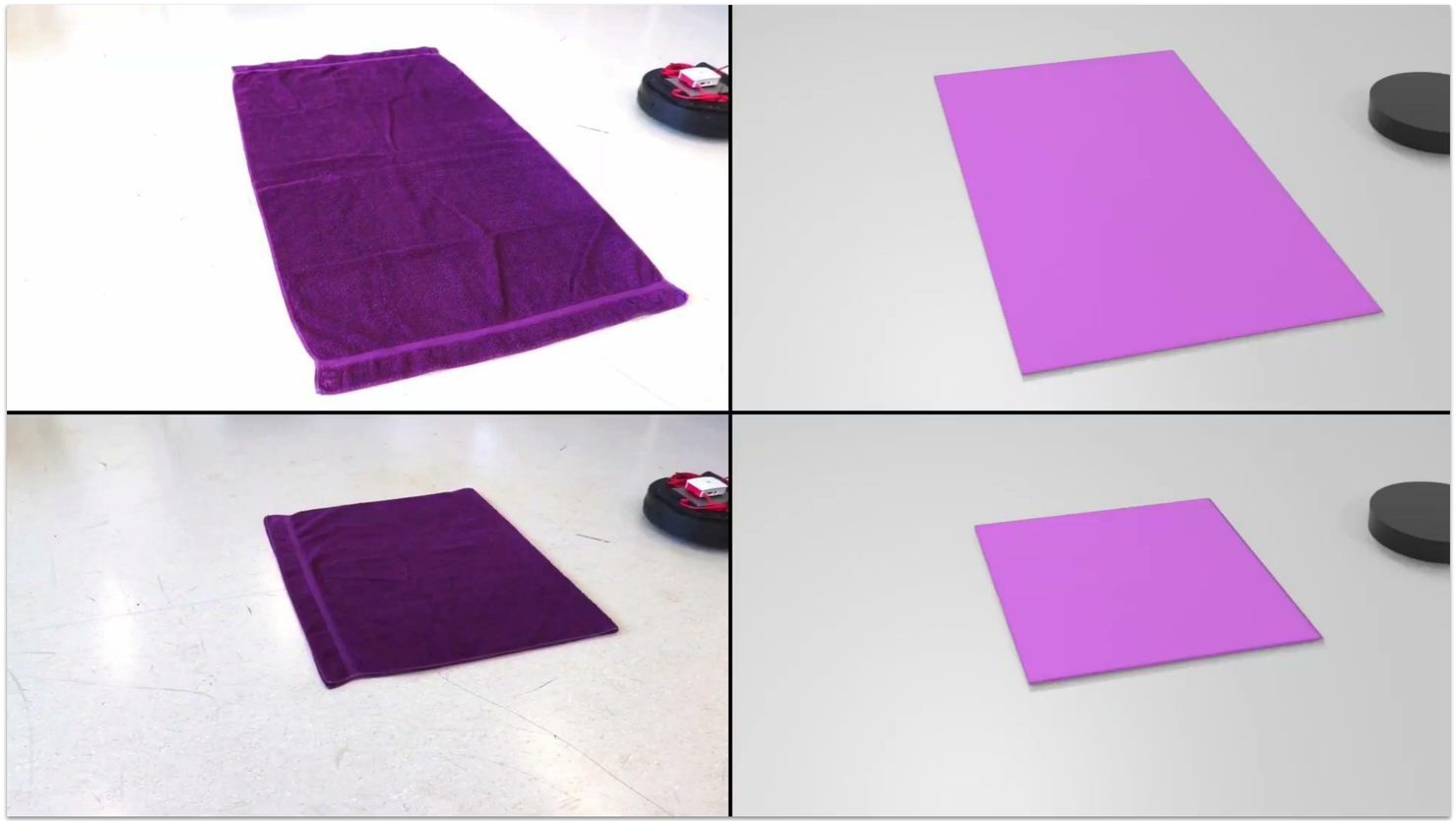


SymX [Fernández-Fernández et al. 2025]

Examples

Strongly Coupled Simulation of Magnetic Rigid Bodies [Westhofen et al. 2024]

Examples



STARK [Fernández-Fernández et al. 2024]

Long History of Optimization in Simulation

ELSEVIER

Comput. Methods Appl. Mech. Engrg. 172 (1999) 203–240

Computer methods in applied mechanics and engineering

Error estimation and adaptive meshing in strongly nonlinear dynamic problems

R. Radovitzky, M. Ortiz*

Graduate Aeronautical Laboratories, California Institute of Technology, Pasadena, CA 9112

Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Geometric, Variational Integrators for Computer Ani

L. Kharevych Weiwei Y. Tong E. Kanso‡ J. E. Marsden P. Schröder M. Desbru

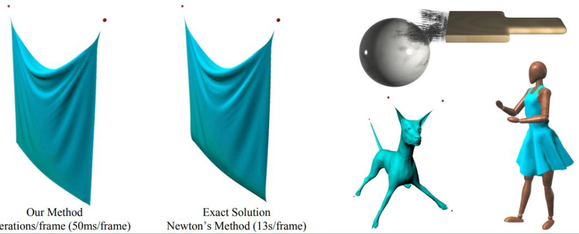
Caltech - ‡USC

Abstract

We present a general-purpose numerical scheme for time integration of Lagr portant computational tool at the core of most physics-based animation tech particular time integrator highly desirable for computer animation: it numeric

Fast Simulation of Mass-Spring Systems

Tiantian Liu, University of Pennsylvania
Adam W. Bargteil, University of Utah
James F. O'Brien, University of California, Berkeley
Ladislav Kavan*, University of Pennsylvania



Our Method
10 iterations/frame (50ms/frame)

Exact Solution
Newton's Method (13s/frame)

Example-Based Elastic Materials

Sebastian Martin[1] Bernhard Thomaszewski[1,2] Eitan Grinspun[3] Markus Gross[1,2]

[1]ETH Zurich [2]Disney Research Zurich [3]Columbia University

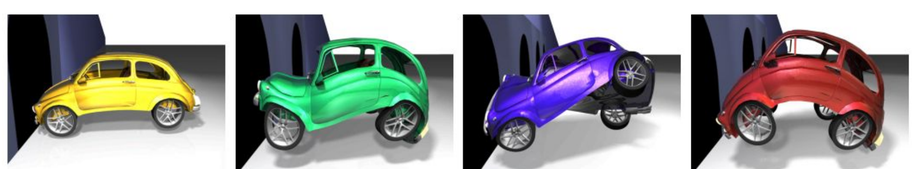


Figure 1: *Example-based materials allow the simulation of flexible structures with art-directable deformation behavior.*

Optimization Integrator for Large Time Steps

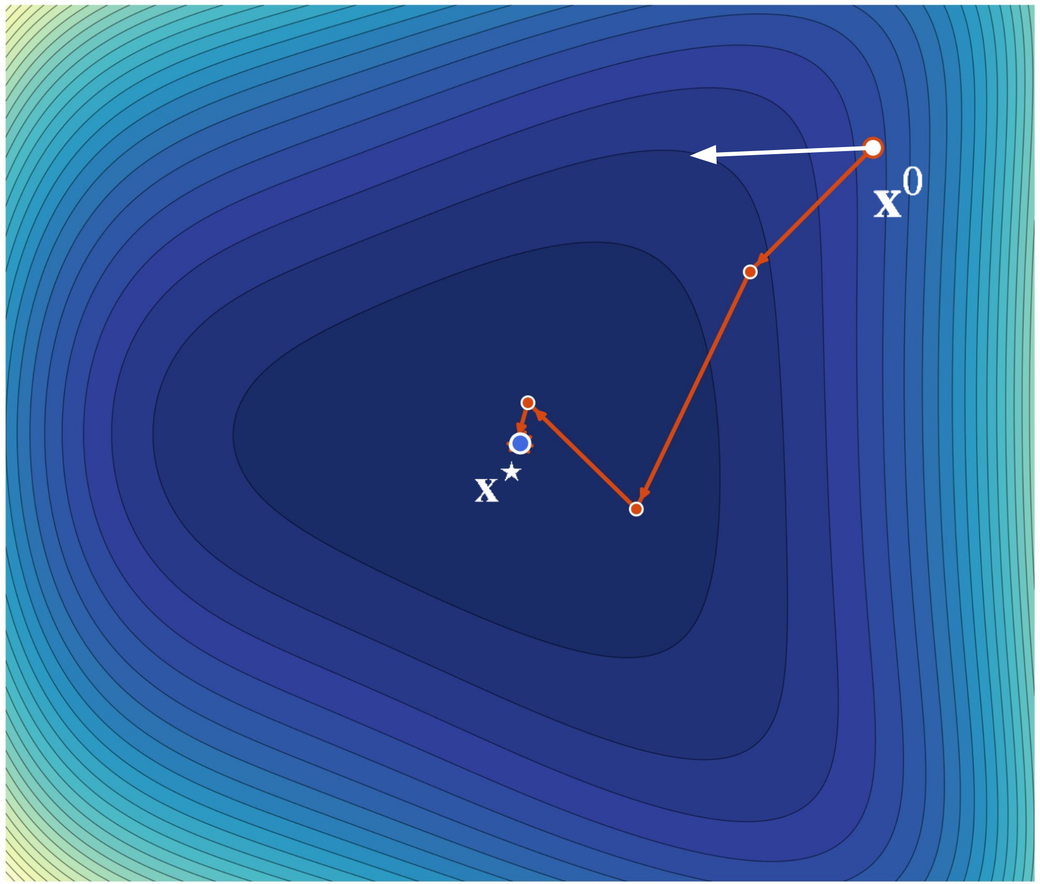
Theodore F. Gast, Craig Schroeder, Alexey Stomakhin, Chenfanfu Jiang and Joseph M. Teran

Abstract—Practical time steps in today's state-of-the-art simulators typically rely on Newton's method to solve large systems of nonlinear equations.

Incremental Potential Contact: Intersection- and Inversion-free, Large-Deformation Dynamics

MINCHEN LI, University of Pennsylvania & Adobe Research
ZACHARY FERGUSON and TESEO SCHNEIDER, New York University
TIMOTHY LANGLOIS, Adobe Research
DENIS ZORIN and DANIELE PANOZZO, New York University
CHENFANFU JIANG, University of Pennsylvania
DANNY M. KAUFMAN, Adobe Research

Let’s Optimize!



Gradient Descent :(

Newton’s Method :D

Recap

- First and foremost: modeling
- These need to hold certain properties
- We need information about them (e.g. derivatives)
- We need proper solver infrastructure

We get robust and unified strong coupling

Foundations

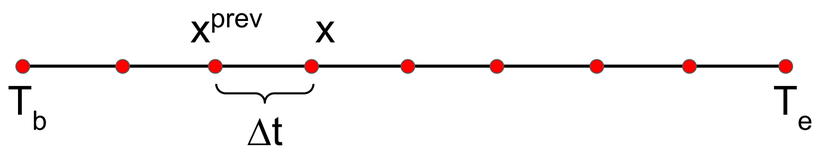
Foundations TOC

1. Optimization Time Integration
2. Newton's Method
3. Global Derivatives and Assembly
4. Penalty and Barrier Constraints
5. Line Search
6. Descent Directions and Hessian Definiteness
7. Convergence and Inexactness

Optimization Time Integration

Equations of Motion: $\mathbf{M}\dot{\mathbf{v}} = \sum_i \mathbf{f}_i(\mathbf{x}, \mathbf{v})$

mass, acc., position, velocity, forces



$$\dot{\mathbf{v}} \approx \frac{\mathbf{v} - \mathbf{v}^{\text{prev}}}{\Delta t}, \quad \mathbf{v} \approx \frac{\mathbf{x} - \mathbf{x}^{\text{prev}}}{\Delta t}$$

(Backward Euler)

Optimization Time Integration

$$\mathbf{M}\frac{\mathbf{x}-\mathbf{x}^{\text{prev}}-\Delta t\mathbf{v}^{\text{prev}}}{\Delta t^2}-\sum_i \mathbf{f}_i(\mathbf{x})=\mathbf{0}$$

→ Root finding

Conservative forces: $\mathbf{f}_i(\mathbf{x})=-\nabla\phi_i(\mathbf{x})$ (potential)

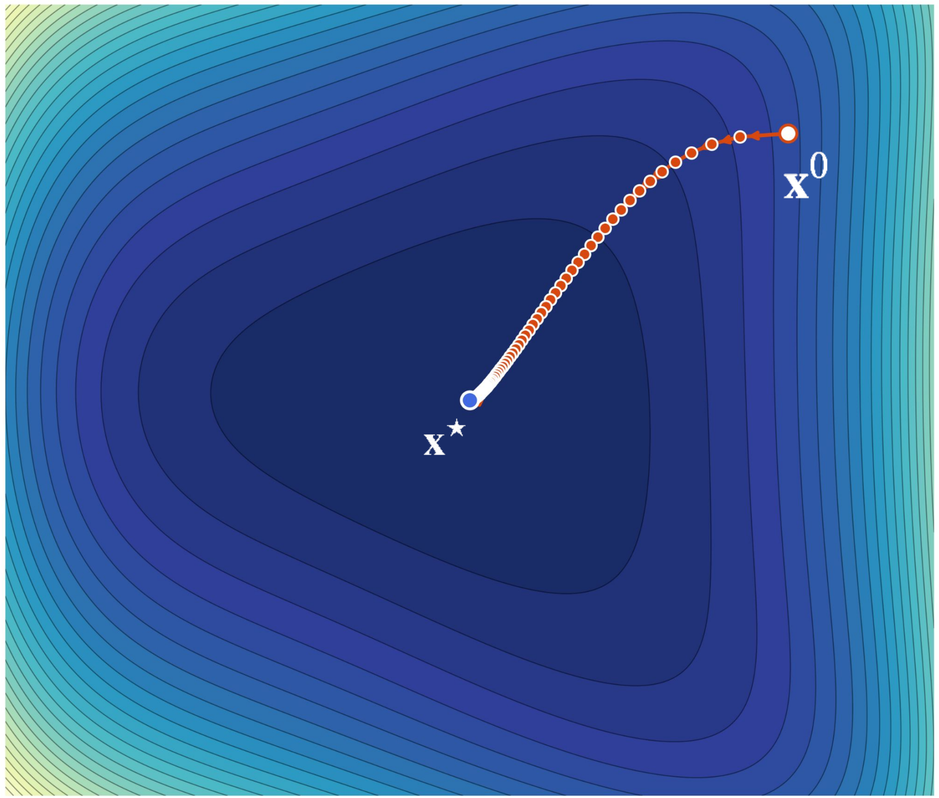
$$\nabla E(\mathbf{x})=\mathbf{0}$$

$$\boxed{E(\mathbf{x})=\frac{1}{2\Delta t^2}(\mathbf{x}-\tilde{\mathbf{x}})^T\mathbf{M}(\mathbf{x}-\tilde{\mathbf{x}})+\sum_i \phi_i(\mathbf{x})}$$

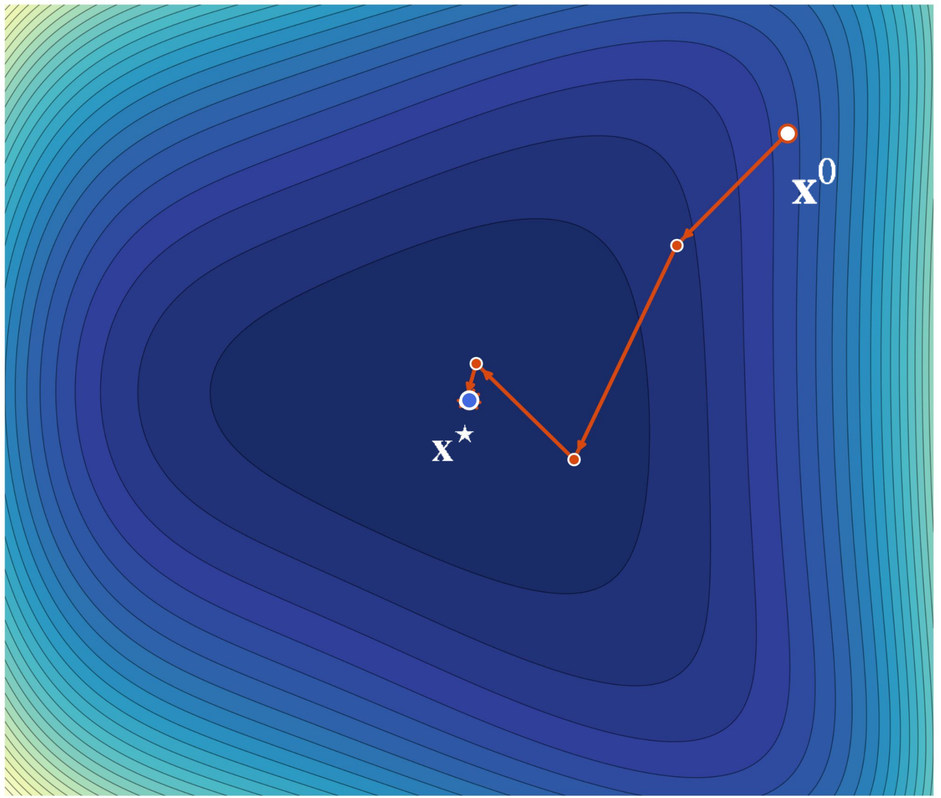
→ Optimization

$$\tilde{\mathbf{x}}=\mathbf{x}^{\text{prev}}+\Delta t\mathbf{v}^{\text{prev}}$$

Newton's Method



Gradient Descent



Newton's Method

But how?

Newton's Method

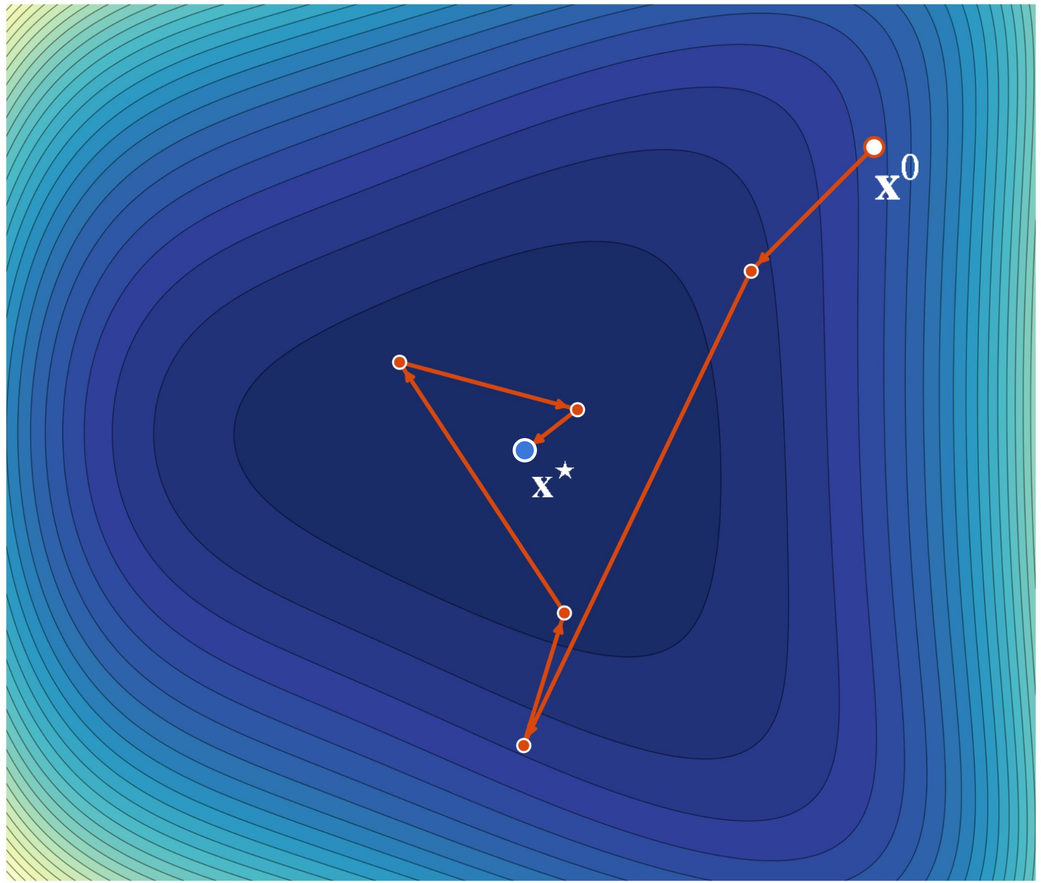
Quadratic model:

$$m_k(\Delta\mathbf{x}) = E(\mathbf{x}) + \nabla E(\mathbf{x})^T \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T \nabla^2 E(\mathbf{x}) \Delta\mathbf{x}$$

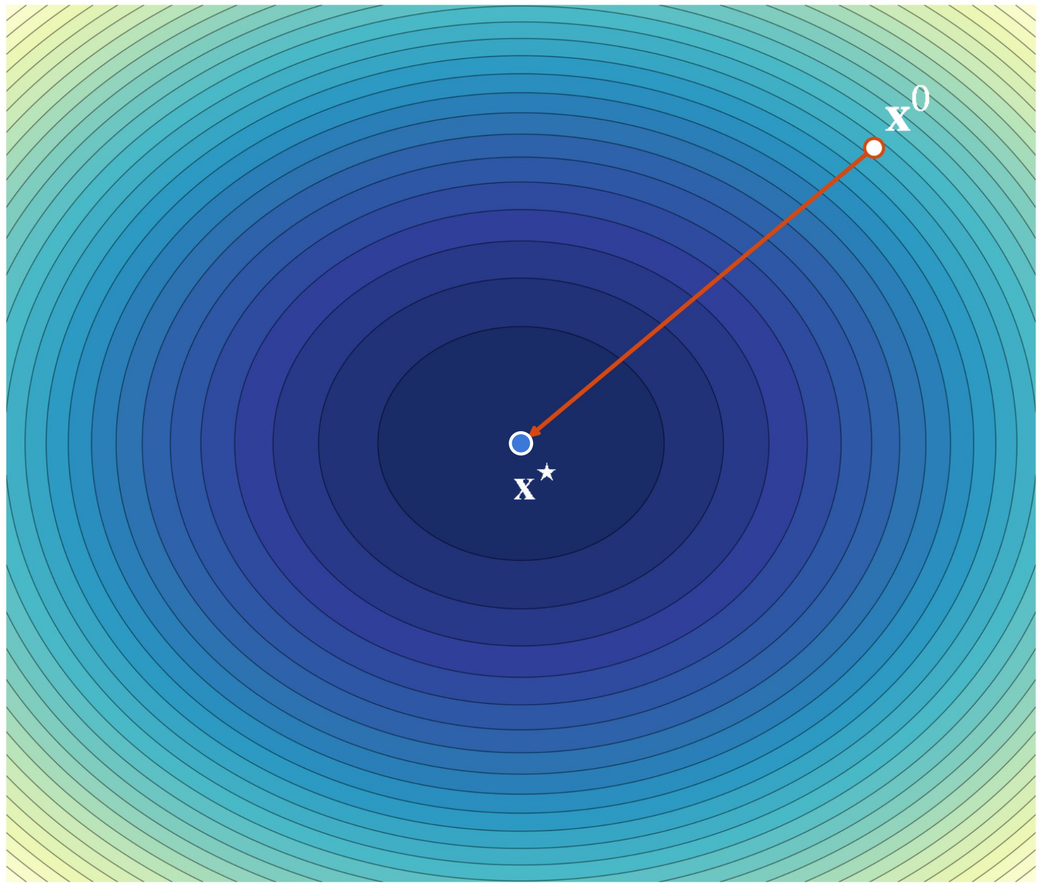
Go to minimizer:

$$\nabla^2 E(\mathbf{x})\, \Delta\mathbf{x} = -\nabla E(\mathbf{x})$$

Newton’s Method

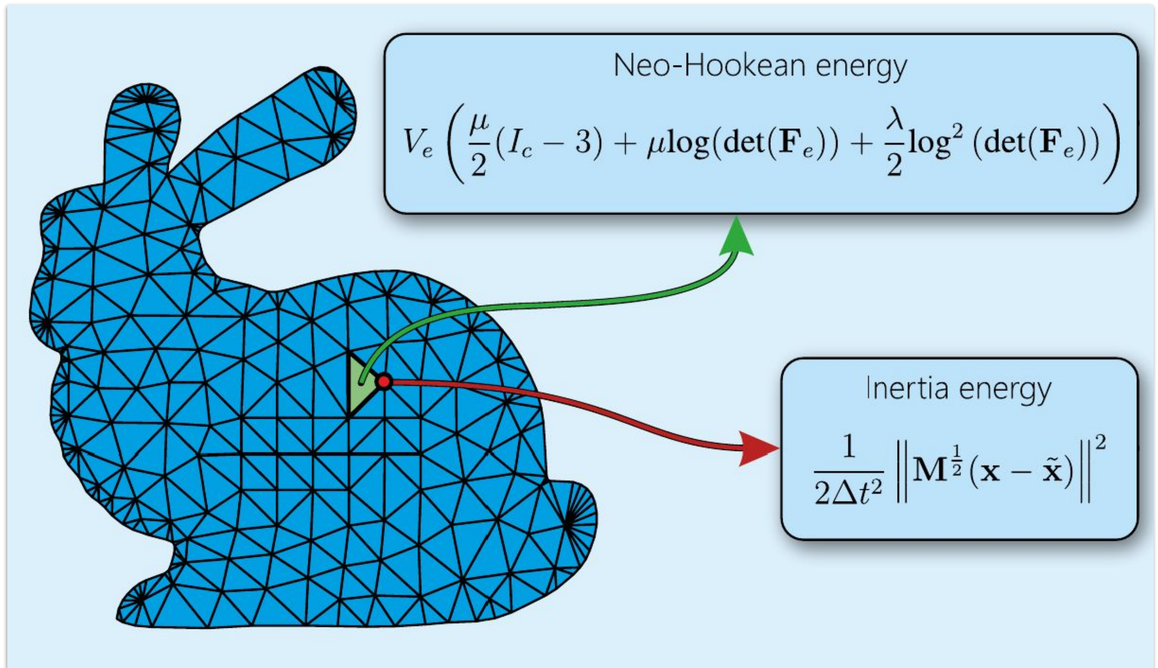


If *E* is quadratic…



1 iteration!

Global Derivatives and Assembly



SymX [Fernández-Fernández et al. 2025]

$\mathcal{P}$ Set of unique potentials

$\mathcal{T}$ Set of discretization elements

1×1

$$E(\mathbf{x}) = \sum_{p\in\mathcal{P}}\sum_{e\in\mathcal{T}_p}\phi_p^e$$

1×1

N×1

$$\mathbf{g} = \nabla E(\mathbf{x}) = \sum_{p\in\mathcal{P}}\sum_{e\in\mathcal{T}_p}(\mathbf{P}^e)^T\nabla\phi_p^e$$

NxM

M×1

N×N

$$\mathbf{H} = \nabla^2 E(\mathbf{x}) = \sum_{p\in\mathcal{P}}\sum_{e\in\mathcal{T}_p}(\mathbf{P}^e)^T\nabla^2\phi_p^e\mathbf{P}^e$$

M×M

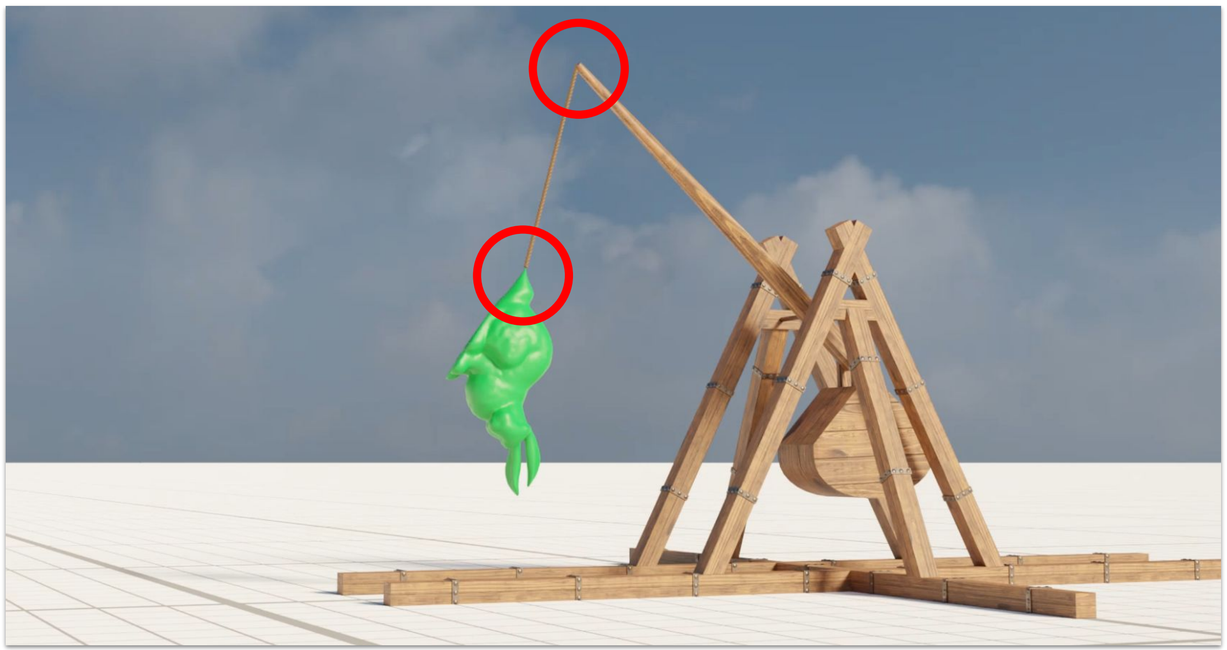
Multi Systems

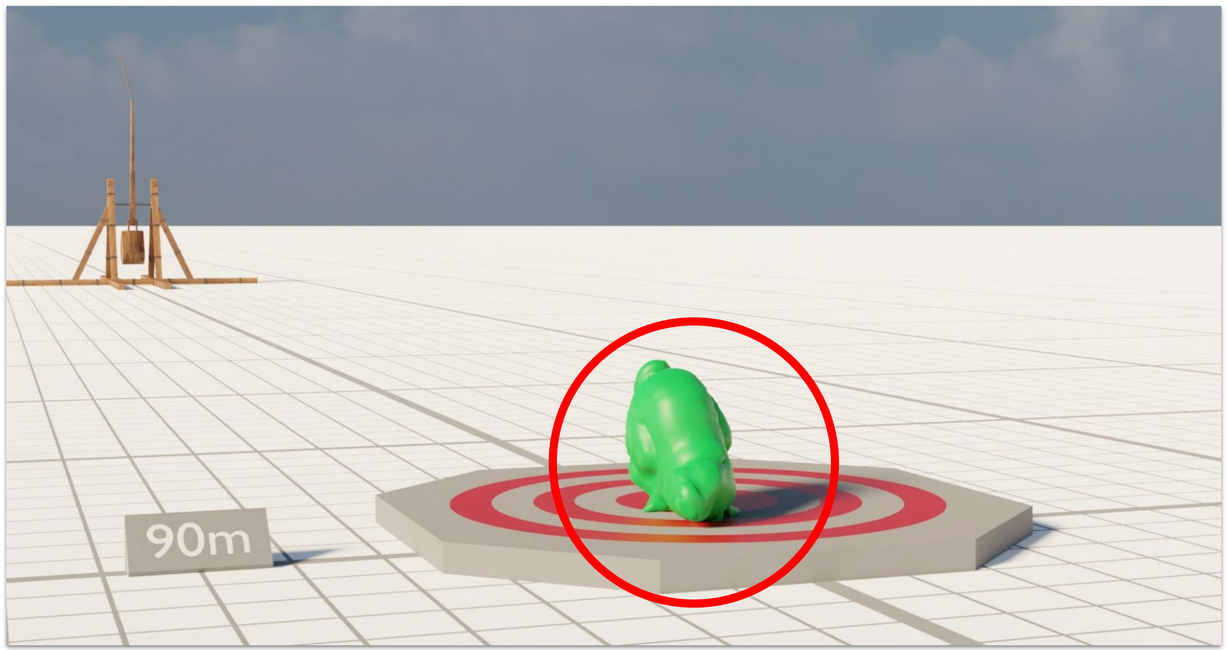
Multiple sets of DoFs:

$$\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_n)^T$$

STARK [Fernández-Fernández et al. 2024]

Penalty and Barrier Constraints





SymX [Fernández-Fernández et al. 2025]

Penalty and Barrier Constraints

Make it expensive:

$$\phi_p(\mathbf{x}) = \frac{1}{2} k_c C(\mathbf{x})^2$$

stiffness >> 0

constraint violation

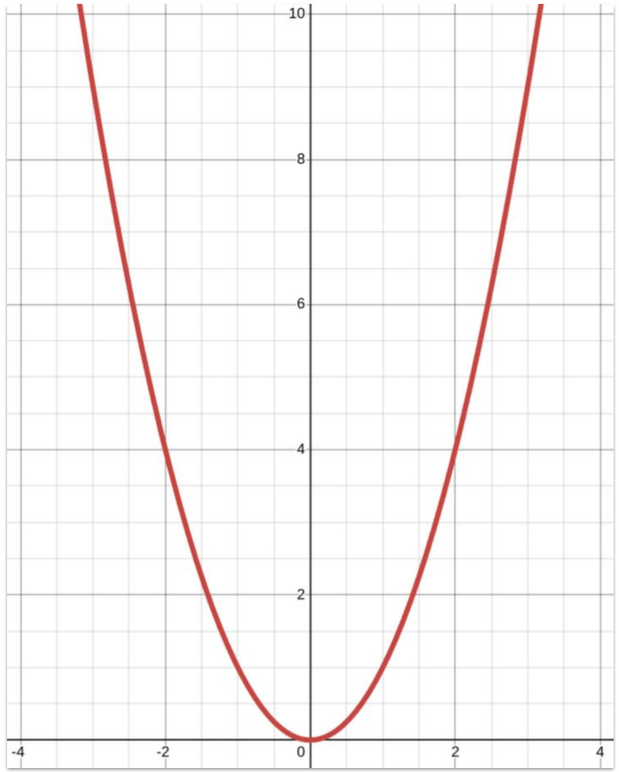
Activation:

$$\phi_{in}(\mathbf{x}) = \begin{cases} 0, & C(\mathbf{x}) \leq 0, \\ \frac{k_c}{3} C(\mathbf{x})^3, & 0 < C(\mathbf{x}) \end{cases}$$

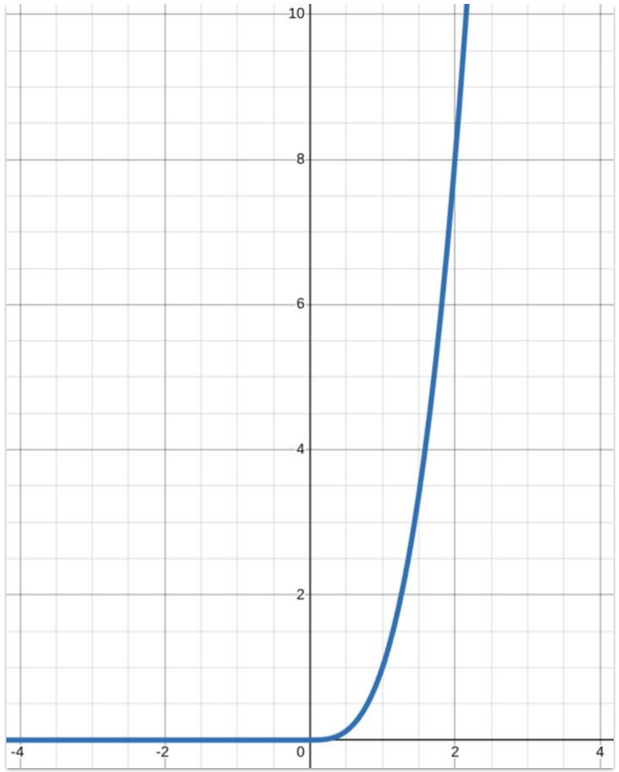
Barrier:

$$\phi_b(\mathbf{x}) = \begin{cases} 0, & C(\mathbf{x}) \leq 0, \\ -k_b C(\mathbf{x})^2 \ln\left(1 - \frac{C(\mathbf{x})}{C_{\text{lim}}}\right), & 0 < C(\mathbf{x}) < C_{\text{lim}}, \\ +\infty, & C(\mathbf{x}) \geq C_{\text{lim}} \end{cases}$$

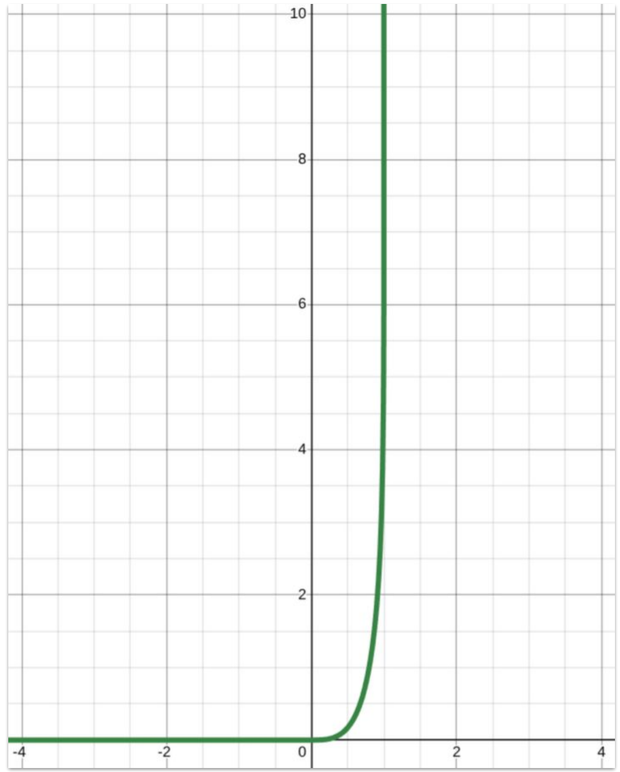
Penalty and Barrier Constraints



Equality Quadratic



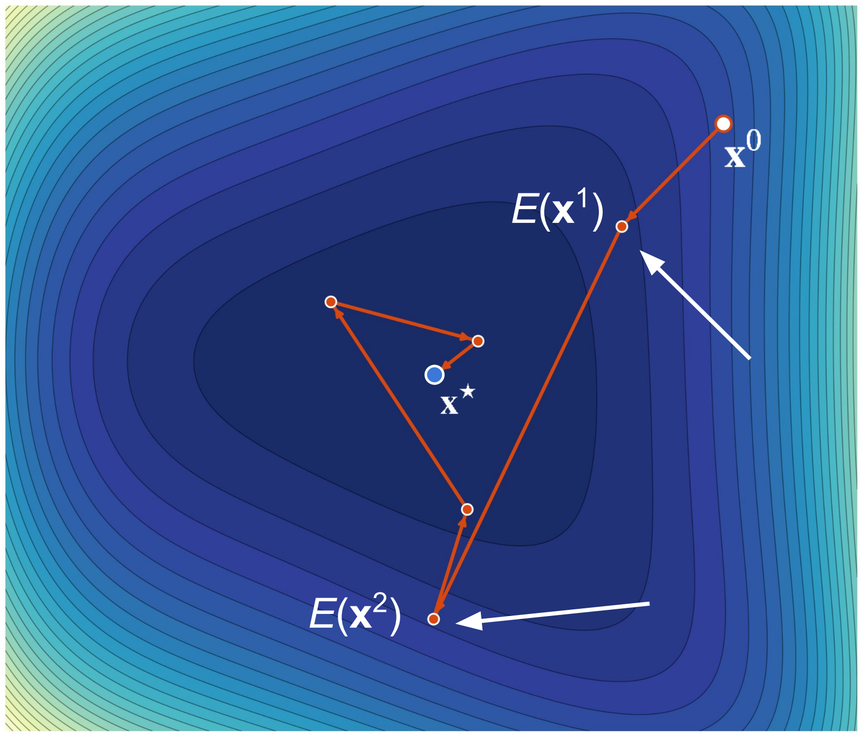
Inequality Cubic



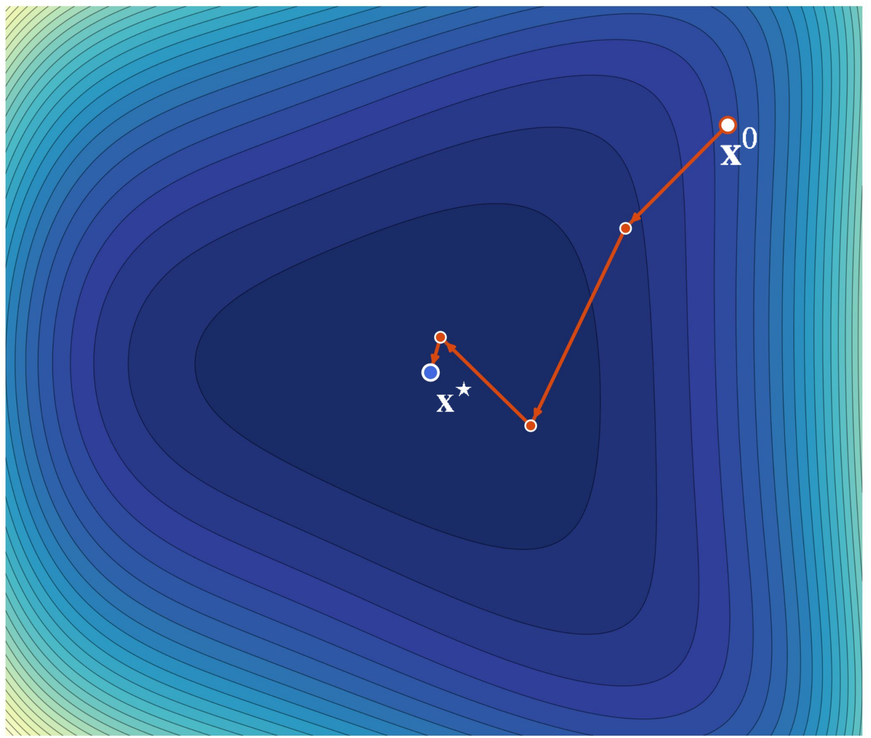
Barrier (C_{lim} = 1)

Line Search

Remember? $E(\mathbf{x}^2) > E(\mathbf{x}^1)$

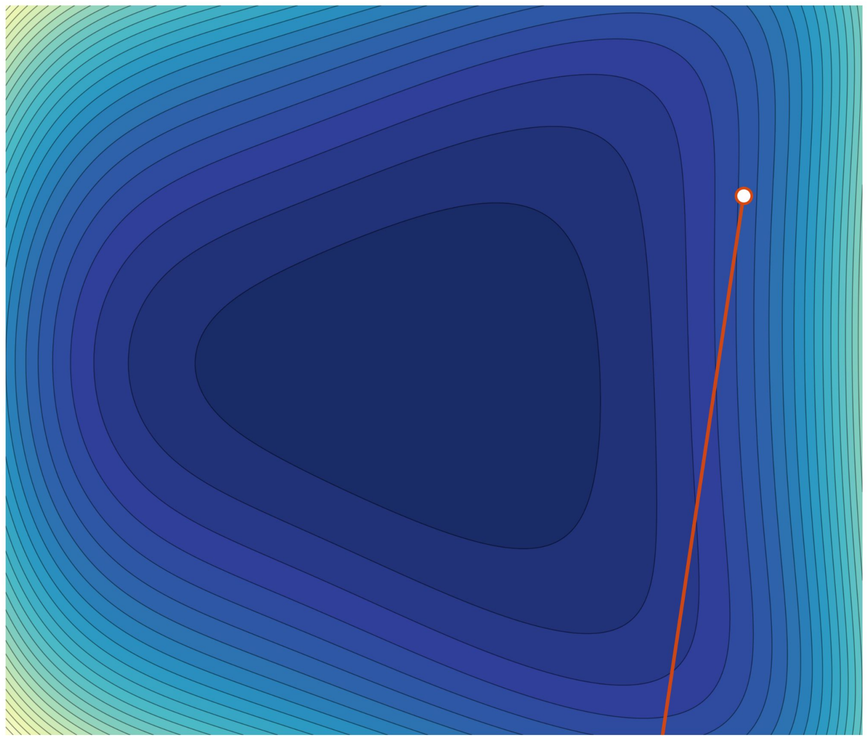


Backtracking!



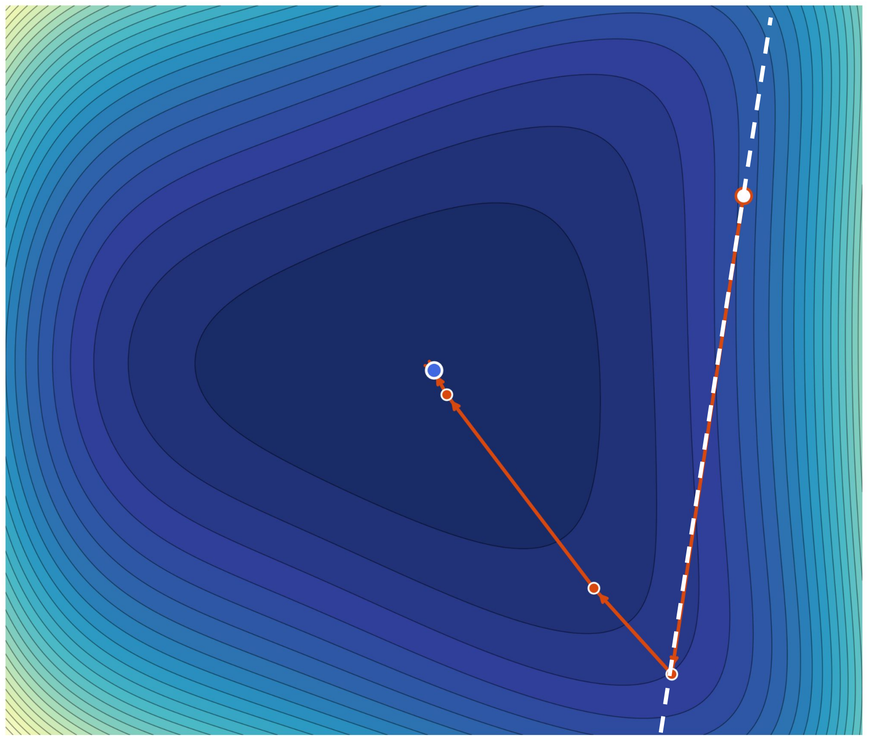
Line Search

Different start

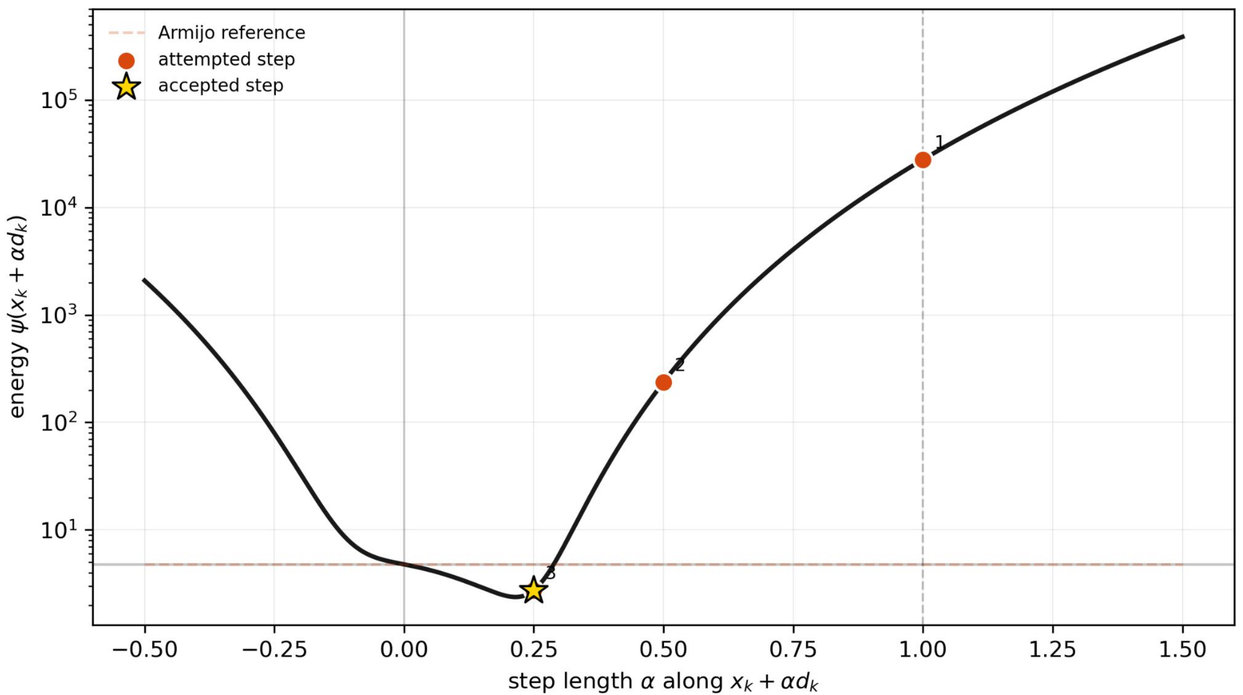


… never to be seen again :(

Backtracking



Line Search



Armijo's Condition

$$E(\mathbf{x} + \alpha\Delta\mathbf{x}) \leq E(\mathbf{x}) + c_1 \underbrace{\alpha \nabla E(\mathbf{x})^T \Delta\mathbf{x}}$$

Tiny fraction

1st order expected decrease

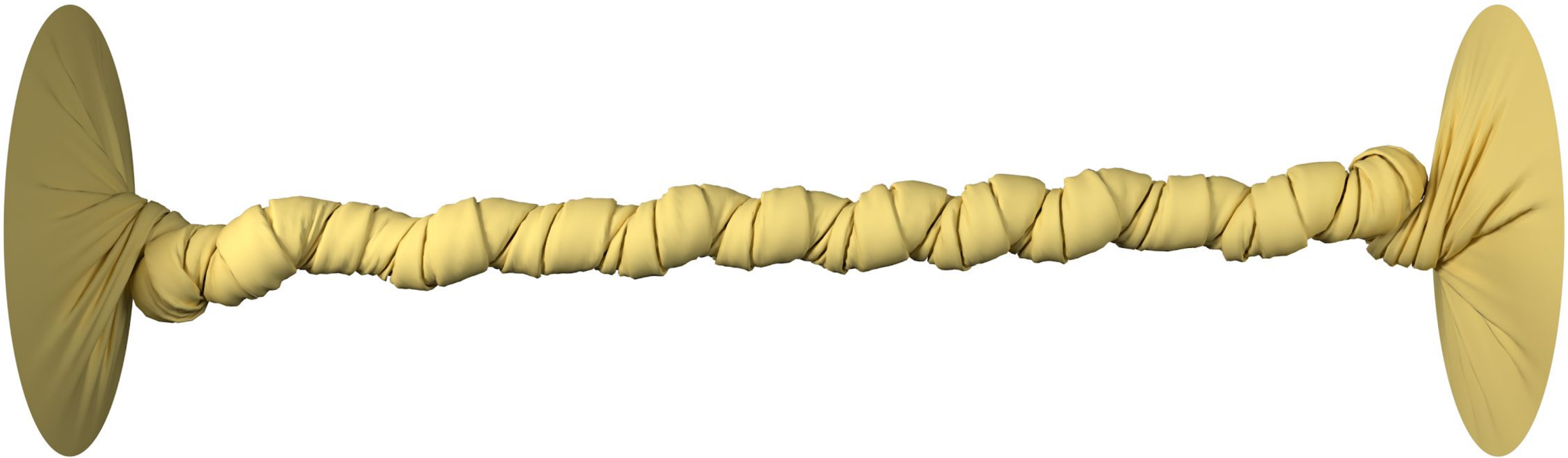
Always Progresses

Line Search

Our line search procedure:

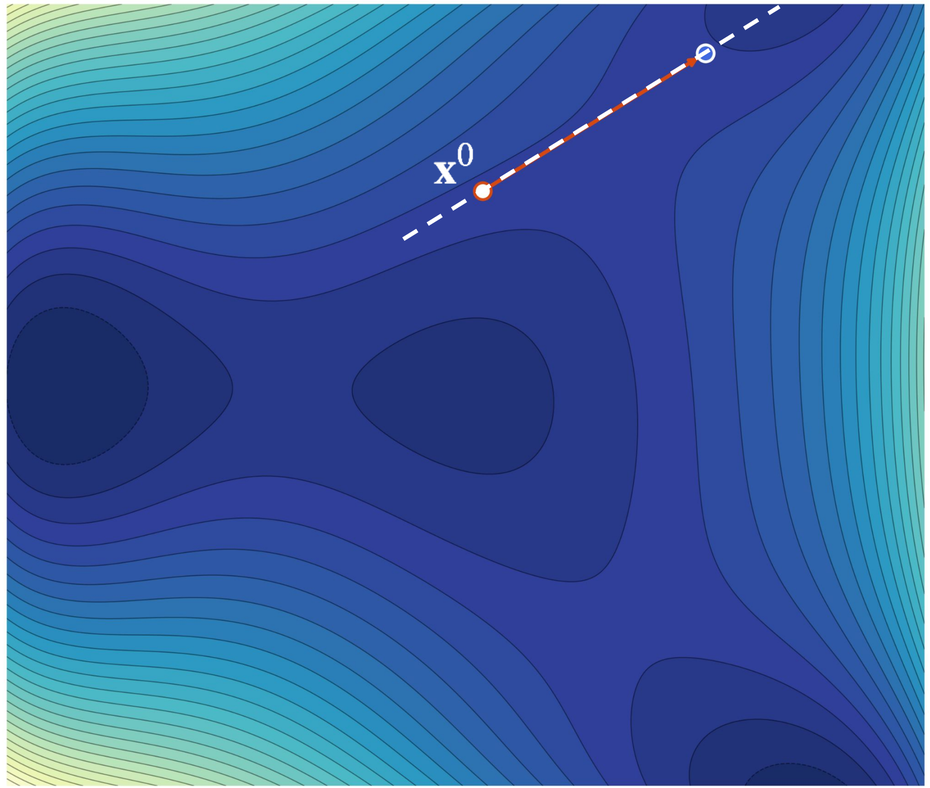
- Step cap
  - Avoids stupidly large steps. Potentially useful for quasistatics.
- Max step
  - Computable max step length in the search direction. E.g. Continuous Collision Detection.
- Admissibility backtracking
  - Backtrack from negative logarithms and sqrts, inverted elements…
- Armijo’s condition backtracking
  - Ensures sufficient descend

Line Search

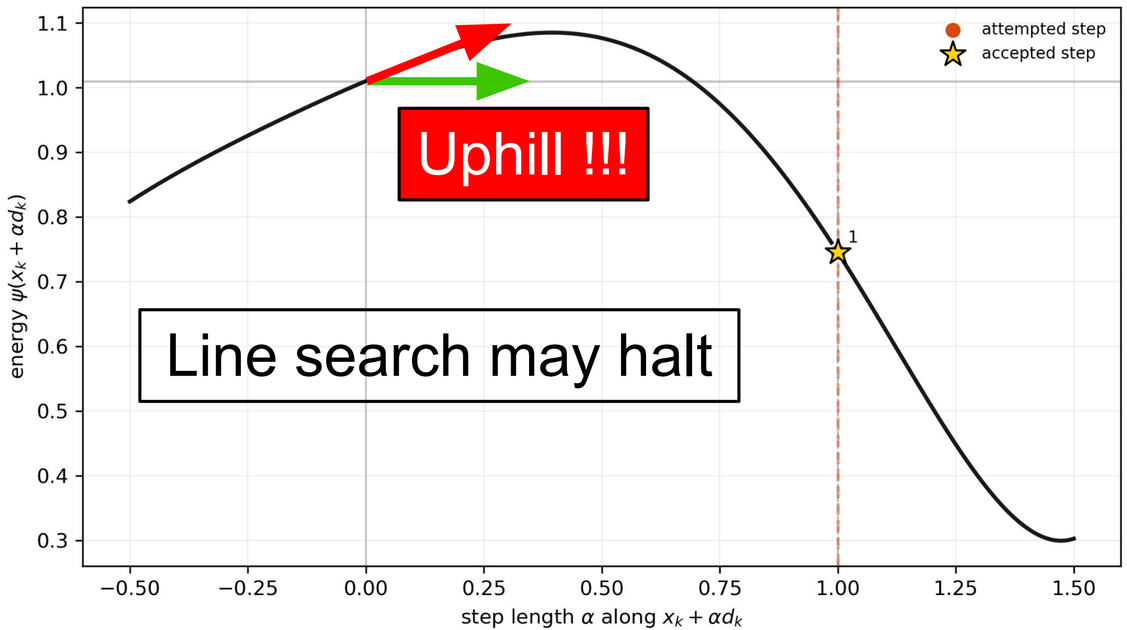


Progressively Projected Newton’s Method [Fernández-Fernández et al. 2026]

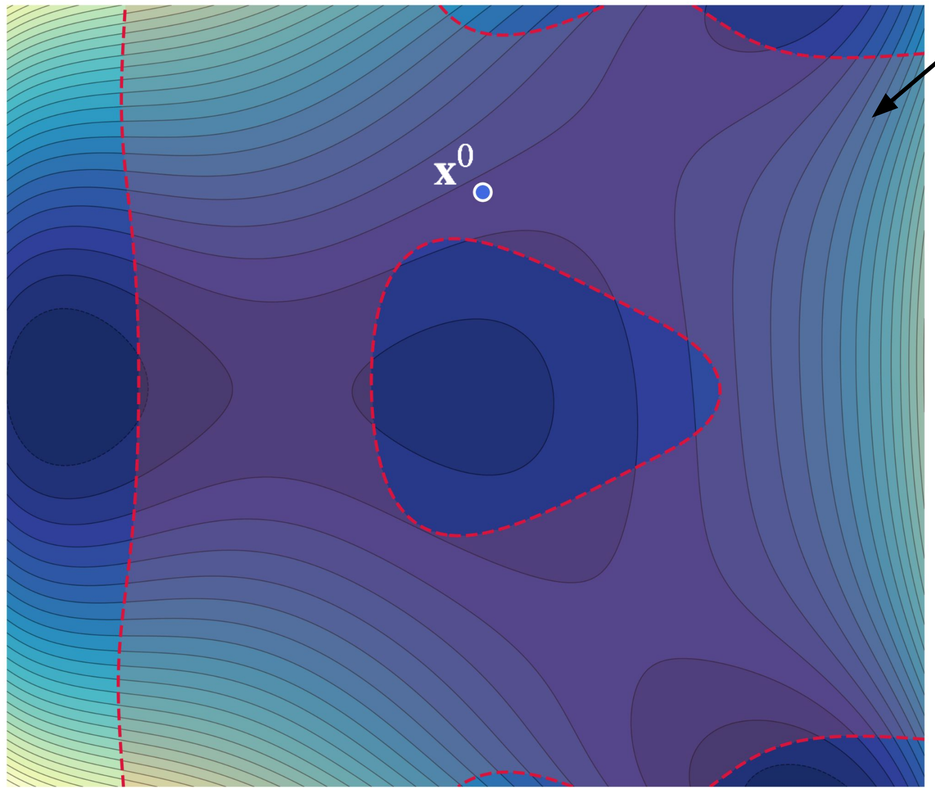
Descent Directions And Hessian Definiteness



First iteration



Descent Directions And Hessian Definiteness



Indefiniteness / Non-convex

We need descent directions: $\Delta\mathbf{x}^T\mathbf{g} < 0$

$$\mathbf{H}\,\Delta\mathbf{x} = -\mathbf{g} \;\rightarrow\; \overbrace{-\underbrace{\Delta\mathbf{x}^T\mathbf{H}\,\Delta\mathbf{x}}_{>0}}^{<0} = \overbrace{\Delta\mathbf{x}^T\mathbf{g}}^{<0} < 0$$

$$\mathbf{p}^T\mathbf{H}\,\mathbf{p} > 0, \quad \forall\,\mathbf{p} \neq \mathbf{0}$$

H is Symmetric Positive (Semi)definite

Descent Directions And Hessian Definiteness

Goal: Non-negative eigvals in **H**

N×N

M×M

$$\mathbf{H}(\mathbf{x}) = \sum_{p \in \mathcal{P}} \sum_{e \in \mathcal{T}_p} \mathbf{P}_e^T \mathbf{H}_p^e(\mathbf{x}) \mathbf{P}_e$$

Projected/Filtered Newton

$$\mathbf{H}_p^e = \mathbf{Q}\,\mathbf{D}\,\mathbf{Q}^T \longrightarrow \widehat{\mathbf{H}}_p^e = \mathbf{Q}\,\widehat{\mathbf{D}}\,\mathbf{Q}^T$$

$$\widehat{D}_{ii}^e = \max(D_{ii}^e, \varepsilon) \quad \text{Clamping}$$

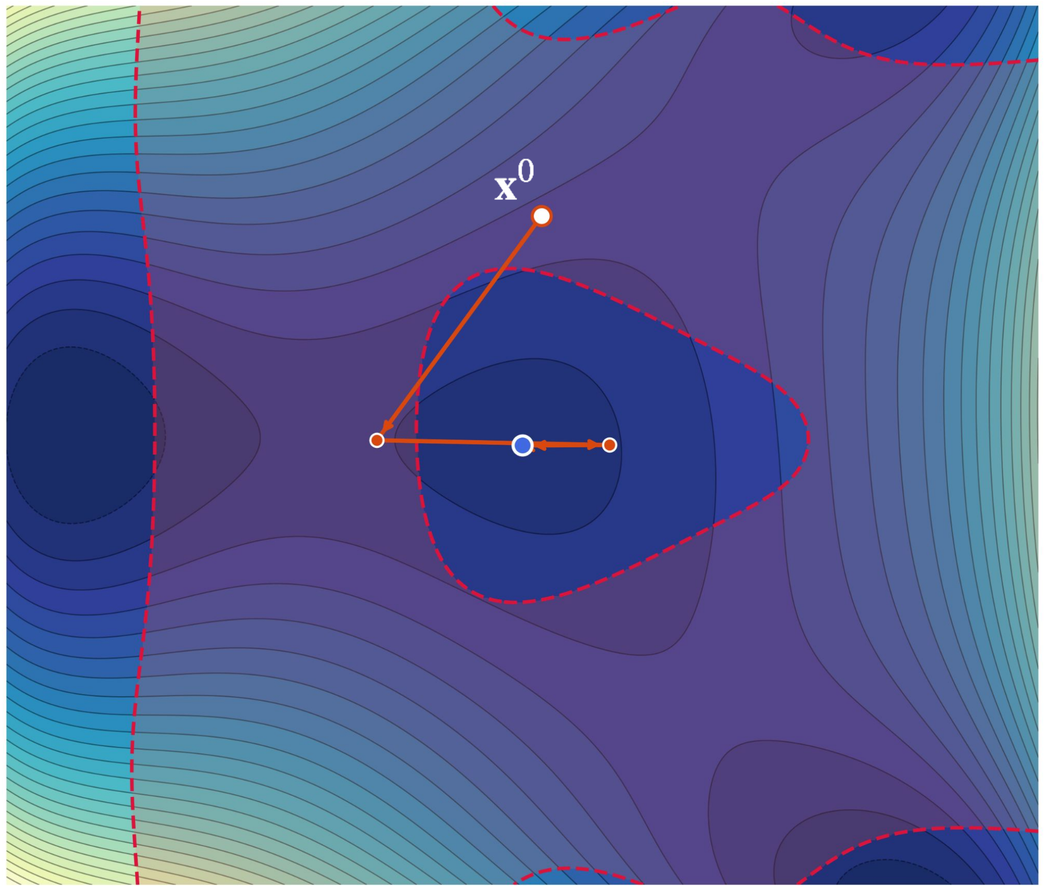
$$\widehat{D}_{ii}^e = \mathrm{abs}(D_{ii}^e) \quad \text{Mirroring}$$

Adaptive Projection

- Project-on-Demand (when needed)
- Progressive (when and where needed)

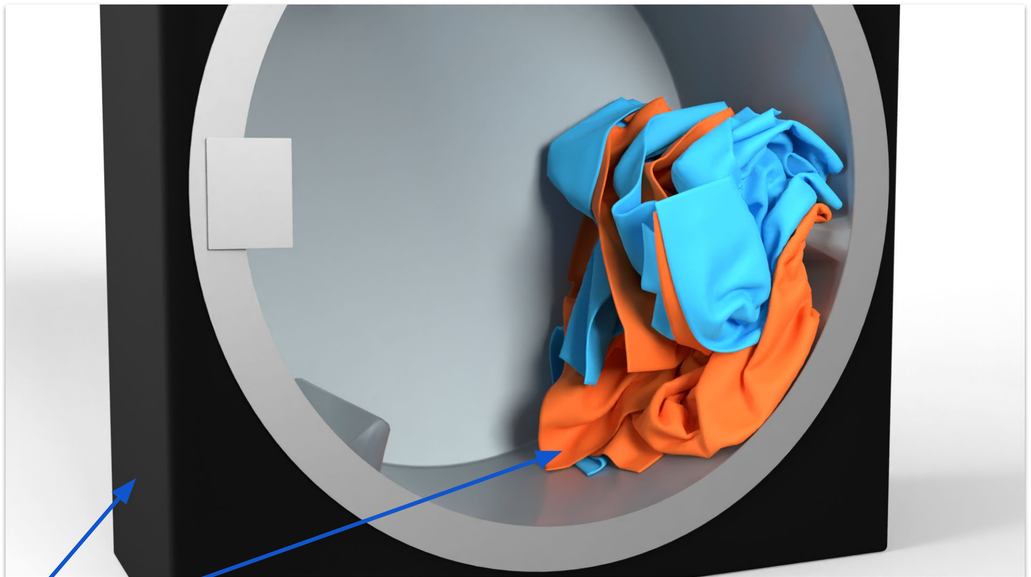
Wednesday · 16:00-17:15 · Kino 5

Descent Directions And Hessian Definiteness



Convergence and Inexactness

- Newton is rarely driven to very high accuracy
- Line search ensures admissible progress
- Common convergence criteria:
  - Fixed Newton iterations
  - Residual norm $\| \nabla E(\mathbf{x}) \| < \varepsilon_{res}$ [N]
  - Step size $\| \Delta \mathbf{x} \| < \varepsilon_{step}$ [m]

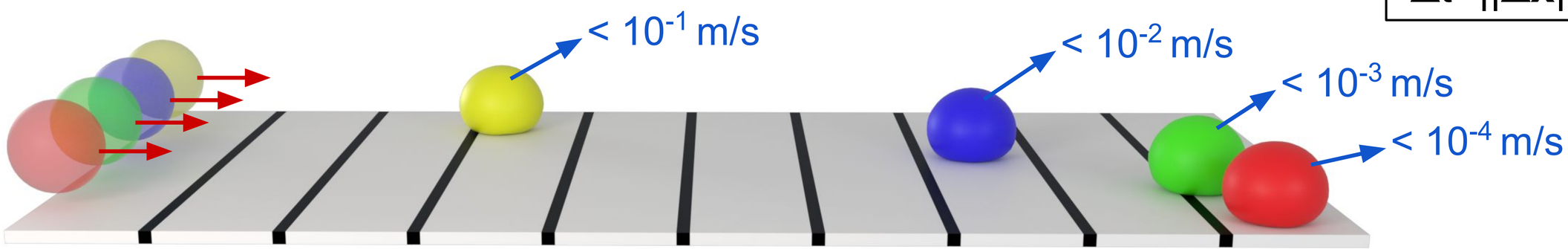


SymX [Fernández-Fernández et al. 2025]

Different residual scales!

Convergence and Inexactness

$\Delta t^{-1}||\Delta \mathbf{x}||_\infty < \text{tol}$

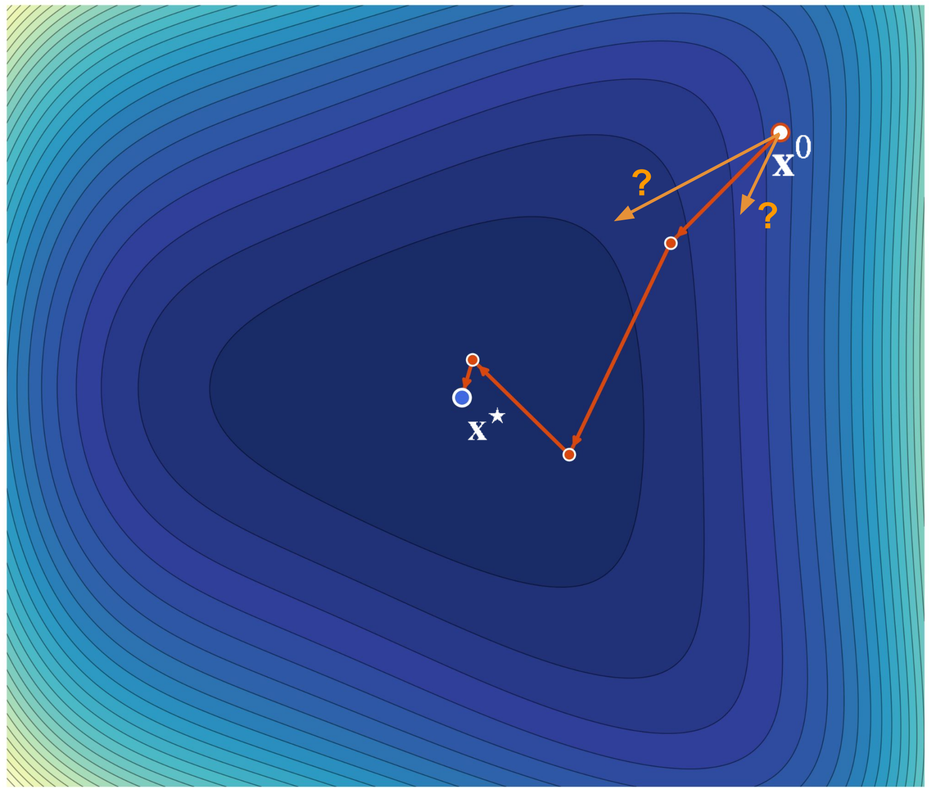


Progressively Projected Newton's Method [Fernández-Fernández et al. 2026]

Consequences of Inexactness:

- Damping
- Overstretching
- Drifting/Sliding

Forcing Sequence



- Iterative linear system solver (PCG)
- Early exits can be very cheap
  - $\varepsilon_{CG} < 10^{-12}$ vs $\varepsilon_{CG} < 10^{-3}$
- Far from $\mathbf{x}^*$ we can be inexact
- Very significant performance boost
- Good practical proxy:

$$\|\mathbf{r}_k\| \leq \eta \|\mathbf{r}_0\|, \quad \eta \approx 10^{-4}$$

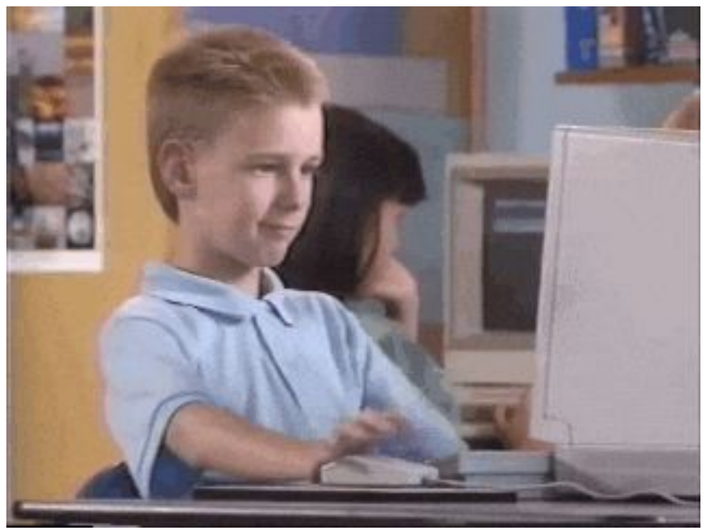
Newton's Procedure

Algorithm 1 Newton's method for optimization of the Backward Euler incremental potential.

$\mathbf{x} \leftarrow \tilde{\mathbf{x}} = \mathbf{x}^{\text{prev}} + \Delta t \mathbf{v}^{\text{prev}}$
while not converged **do**
    assemble gradient $\mathbf{g}(\mathbf{x})$ and Hessian $\mathbf{H}(\mathbf{x})$
    $\hat{\mathbf{H}} \leftarrow \text{project}(\mathbf{H})$
    solve $\hat{\mathbf{H}} \Delta \mathbf{x} = -\mathbf{g}$
    find step size α by line search
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha \Delta \mathbf{x}$
end while
$\mathbf{v} \leftarrow (\mathbf{x} - \mathbf{x}^{\text{prev}})/\Delta t$

Recap

1. Optimization Time Integration
2. Newton's Method
3. Global Derivatives and Assembly
4. Penalty and Barrier Constraints
5. Line Search
6. Descent Directions and Hessian Definiteness
7. Convergence and Inexactness



Later!

SymX: Symbolic Differentiation for Nonlinear Optimization

README Apache-2.0 license

SymX

{SymX}

Symbolic differentiation. C++ code generation. JIT compilation. Global assembly. Non-linear optimization.

Docs · PDF · ACM Page

SymX is a C++ library for **symbolic differentiation** with automatic **code generation**, **compilation** and **evaluation**. Write complex mathematical expressions concisely, differentiate them arbitrarily, and let SymX evaluate them on your data structures — including global gradient and Hessian assembly.

SymX targets **non-linear optimization pipelines** typical of **FEM solvers**, but it can be used for any application that needs JIT compiled math. It uses a **stencil-based** perspective: expressions are defined per *element* and evaluated over a discretization. SymX is the core engine of STARK, a simulation framework for FEM elasticity, shells, rigid bodies, and frictional contact.

Any questions?

SymX Examples

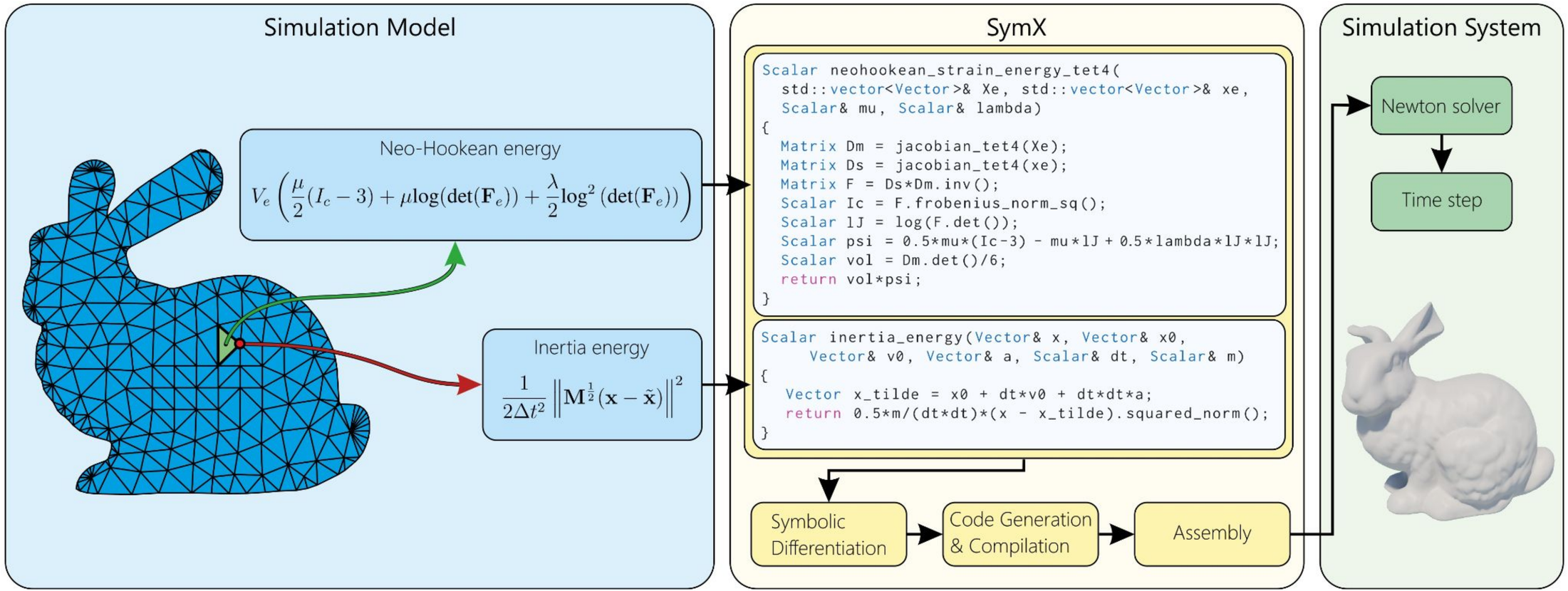
Where to find it

github.com/InteractiveComputerGraphics/SymX

https://symx.physics-simulation.org/

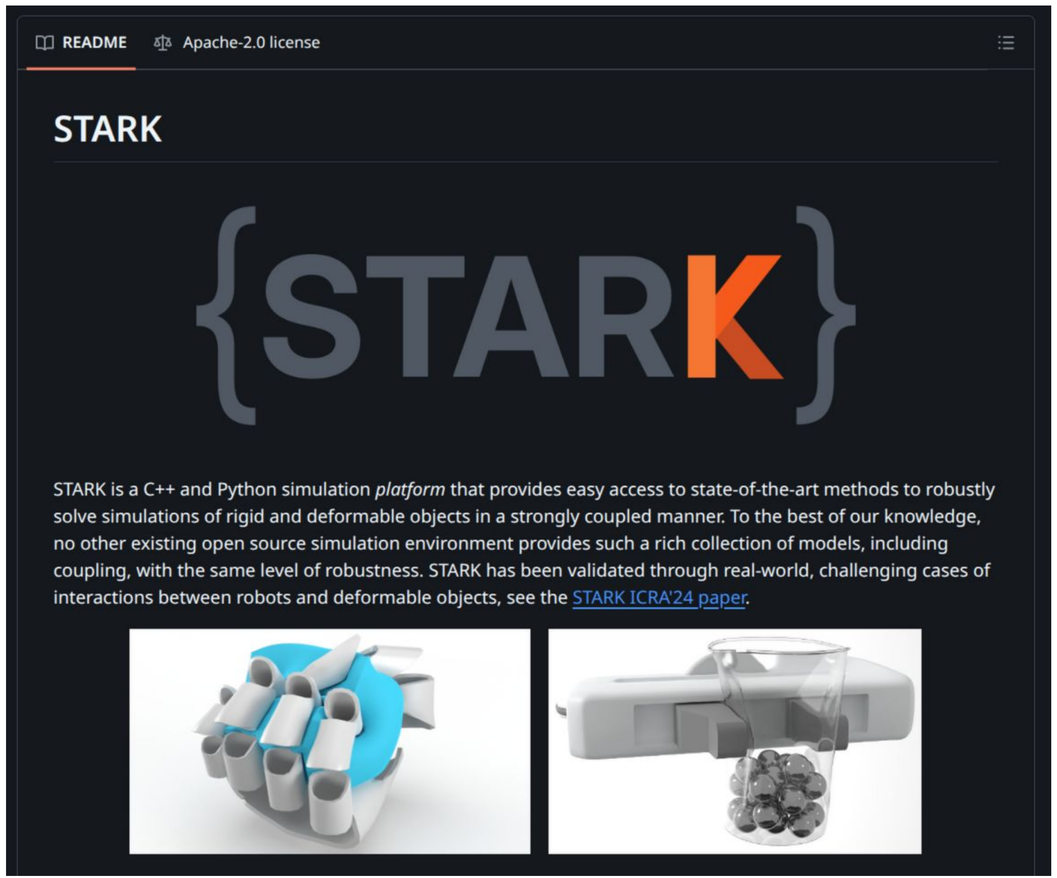
What is SymX?

Symbolic Math Engine



SymX [Fernández-Fernández et al. 2025]

STARK



https://github.com/InteractiveComputerGraphics/stark

Simulation Platform

- Model repository
  - Deformables
  - Rigid Bodies
  - Frictional Contact
- Scripting
- Python API

Example: Cloth Simulation