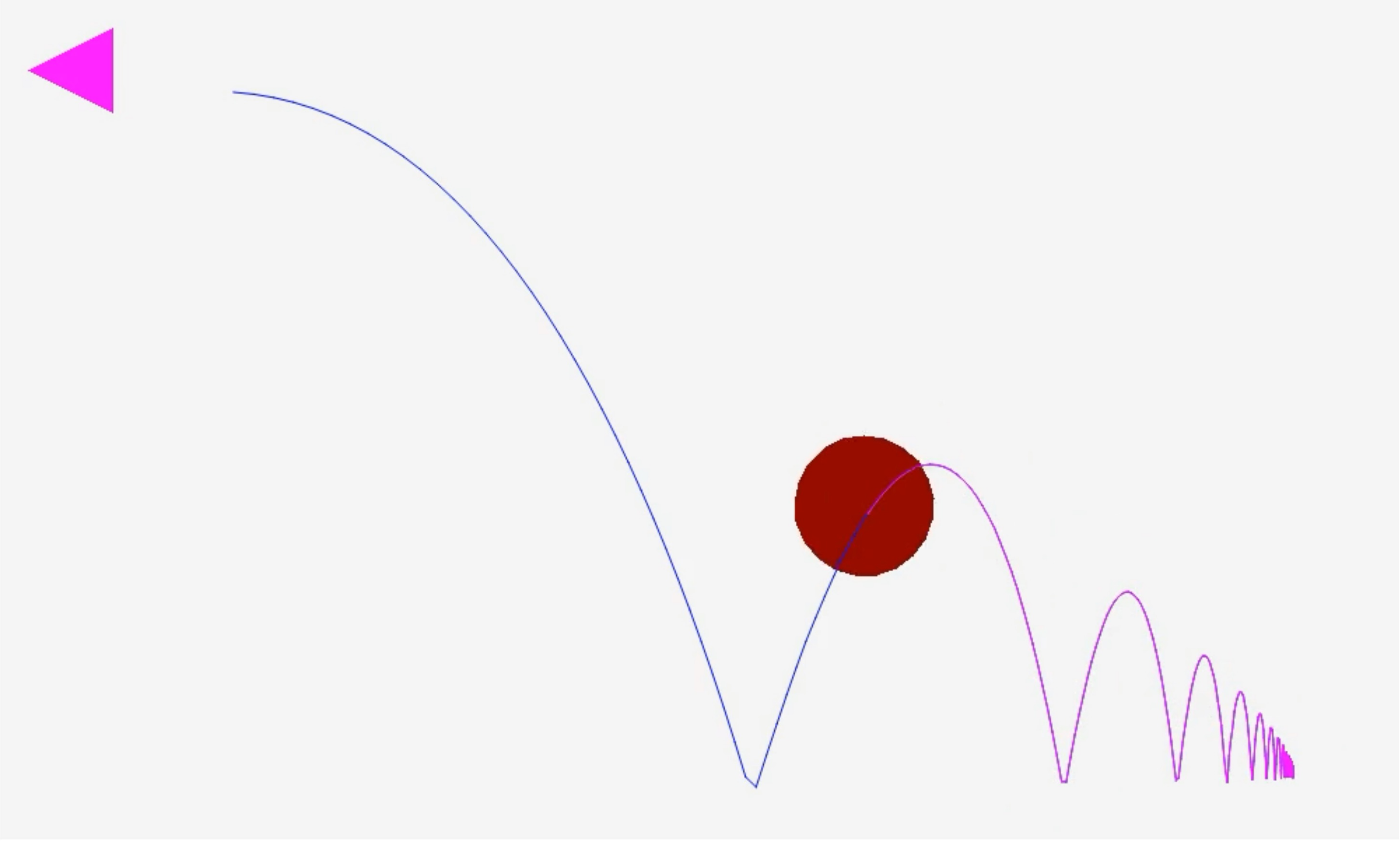
Exactly Bit-Reversible Computational Methods for Memory-Efficient Adjoint Sensitivity Analysis of Dissipative Dynamic Systems

Brian Doran Giffin
Assistant Professor
Structural Engineering & Mechanics
Oklahoma State University

July 20-24, 2025

18th U.S. National Congress on Computational Mechanics

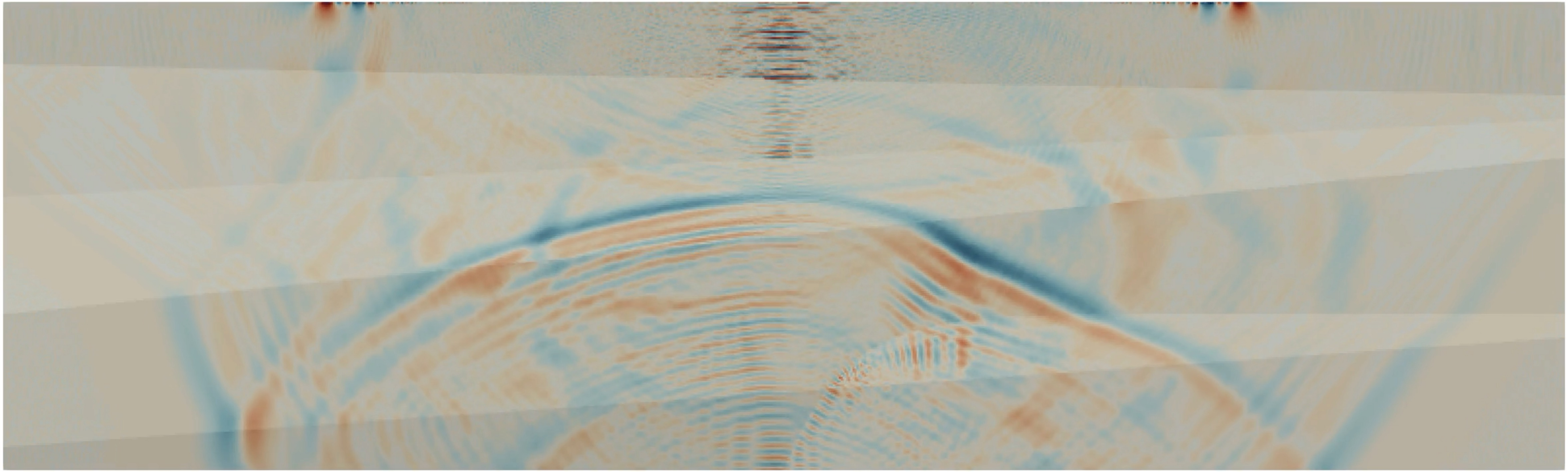
Computational Methods for Inverse Problems and Optimal Experimental Design



The adjoint state method provides an accurate and efficient means of computing sensitivities for dynamic optimization and inverse problems

Forward problem

t



$-t$

Adjoint problem

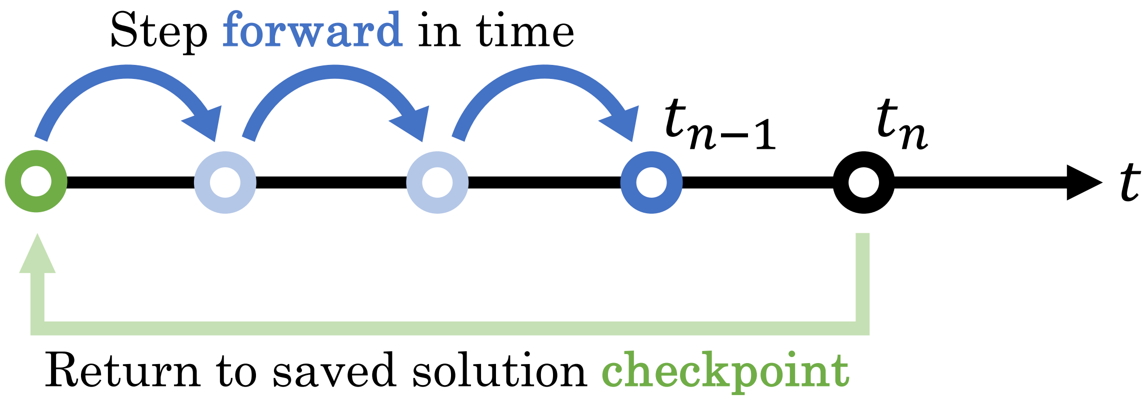
Common implementations of the discrete adjoint method use checkpointing to rematerialize forward solution states necessary for backpropagation

Solution of the adjoint problem must:

- *Store the forward solution* at all prior time states (more memory), or …
- *Rematerialize forward solution* from "checkpoints" (more computations)

Forward update:

$$\boldsymbol{u}_n = \boldsymbol{f}(\boldsymbol{u}_{n-1})$$



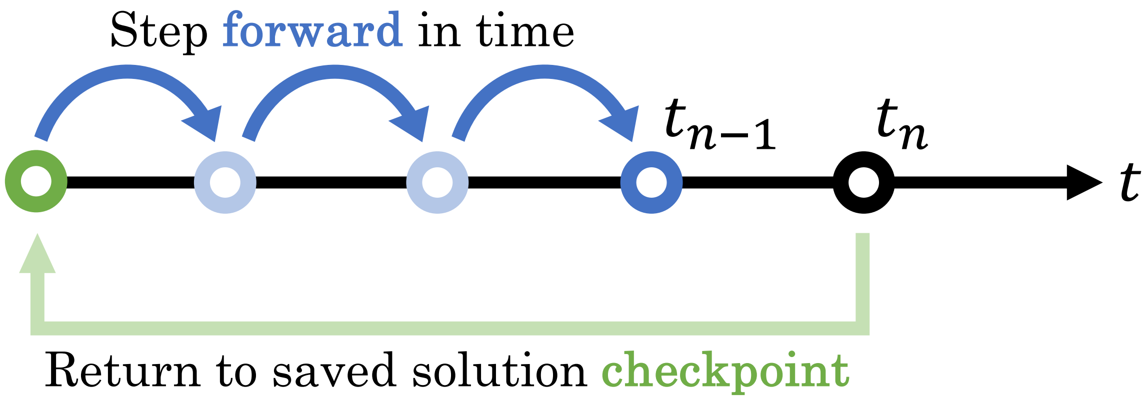
Common implementations of the discrete adjoint method use checkpointing to rematerialize forward solution states necessary for backpropagation

Solution of the adjoint problem must:

- *Store the forward solution* at all prior time states (more memory), or …
- *Rematerialize forward solution* from “checkpoints” (more computations)

Forward update:

$$\boldsymbol{u}_n = \boldsymbol{f}(\boldsymbol{u}_{n-1})$$



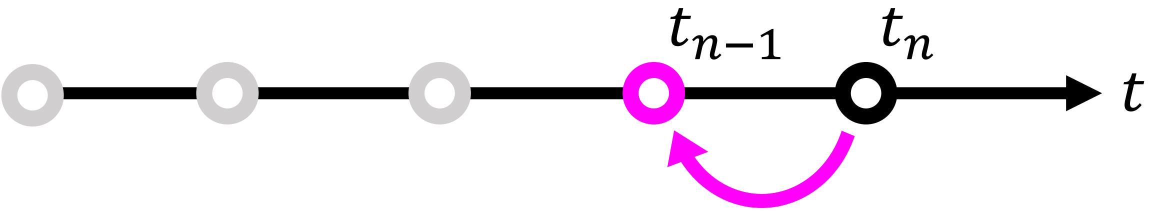
What if we could go *backwards* in time?

Hypothetical solution:

- *Reverse the computations*,
  and recover forward solution at each preceding step
- Requires *minimal memory and recomputation*

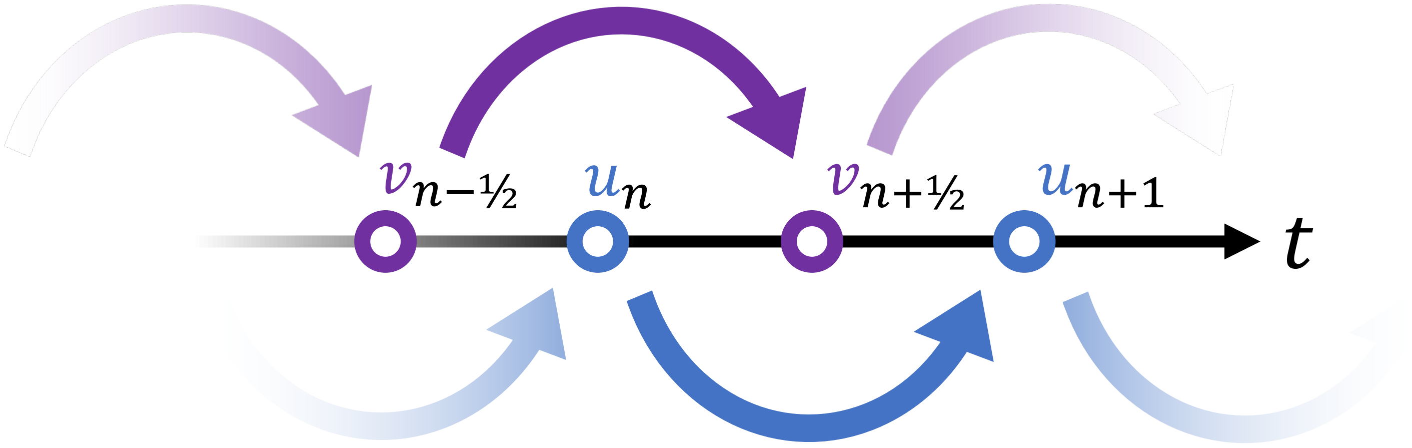
Backward update:

$$\boldsymbol{u}_{n-1} = \boldsymbol{f}^{-1}(\boldsymbol{u}_n)$$



Reverse computations to
step **backwards** in time

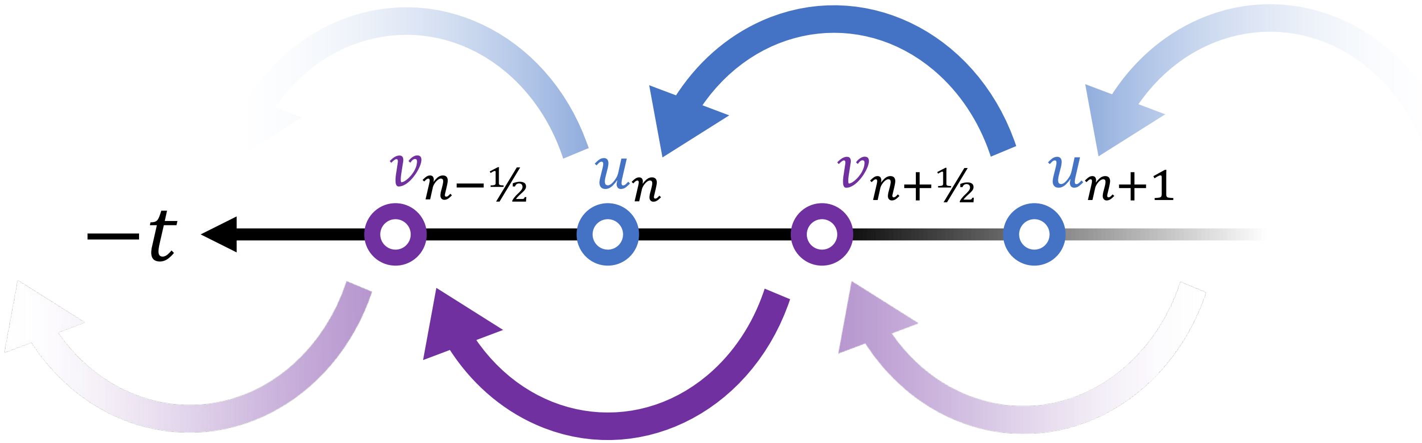
The leapfrog time-integrator is trivially reversible for Hamiltonian systems



Forward update:

$$v_{n+½} \leftarrow v_{n-½} + a(u_n)\Delta t$$

$$u_{n+1} \leftarrow u_n + v_{n+½}\Delta t$$



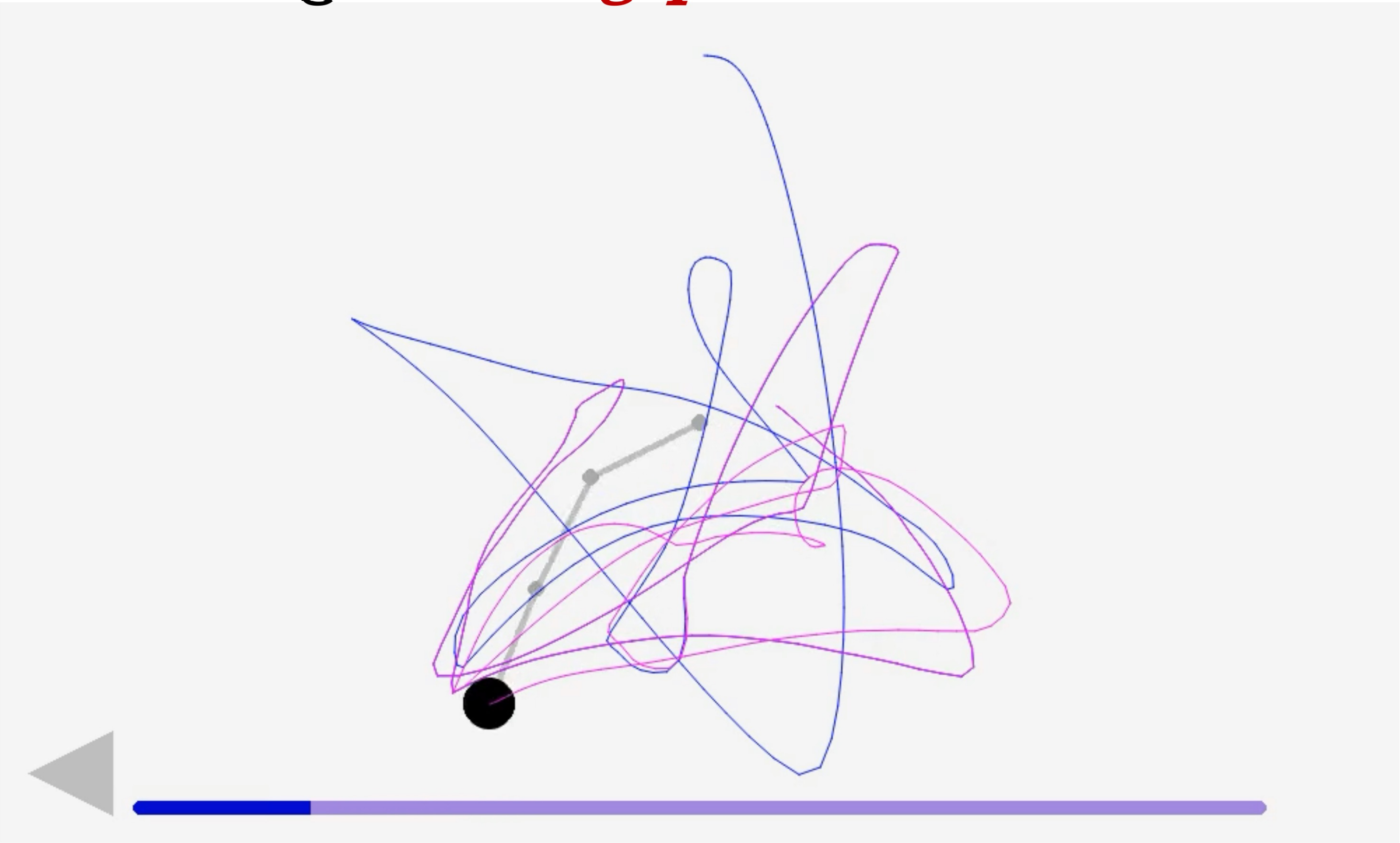
Backward update:

$$u_n \leftarrow u_{n+1} - v_{n+½}\Delta t$$

$$v_{n-½} \leftarrow v_{n+½} - a(u_n)\Delta t$$

Naïve time-reversal leads to inexact rematerialization due to round-off errors from *floating-point arithmetic*

Using *floating-point arithmetic*:



Forward update:

$$v_{n+1/2} \leftarrow v_{n-1/2} + a(u_n)\Delta t$$

$$u_{n+1} \leftarrow u_n + v_{n+1/2}\Delta t$$

Backward update:

$$u_n \leftarrow u_{n+1} - v_{n+1/2}\Delta t$$

$$v_{n-1/2} \leftarrow v_{n+1/2} - a(u_n)\Delta t$$

Represent displacement and velocity degrees of freedom as fixed-width (32- or 64-bit) integers with implied (but differing) problem-dependent radices

mantissa radix exponent

Fixed $x \in \mathbb{R}$ $\quad$ $x = m \times \underbrace{R^e}_{\textit{fixed}}$ $\quad$ **Integer** $m \in \mathbb{Z}$

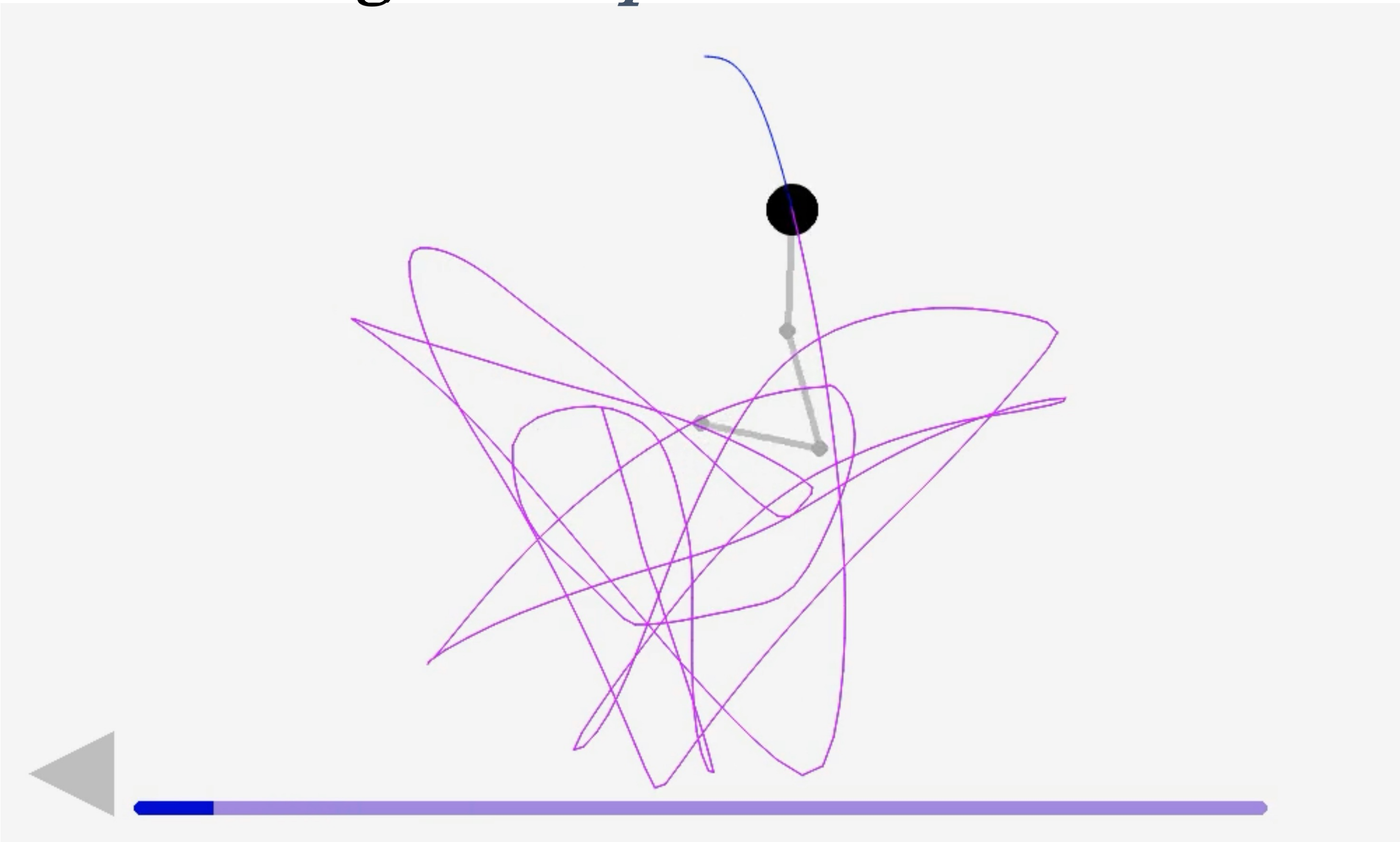
e.g. $\quad 7.560239 = 7560239 \times 10^{-6}$

$x \in [-2147.483648, +2147.483647]$ (32-bit int)

$x \in [-9.223 \ldots \times 10^{12}, +9.223 \ldots \times 10^{12}]$ (64-bit int)

Round-off errors from addition/subtraction can be eliminated using *fixed-point arithmetic*, resulting in exactly "bit-reversible" time-integration

Using *fixed-point arithmetic*:

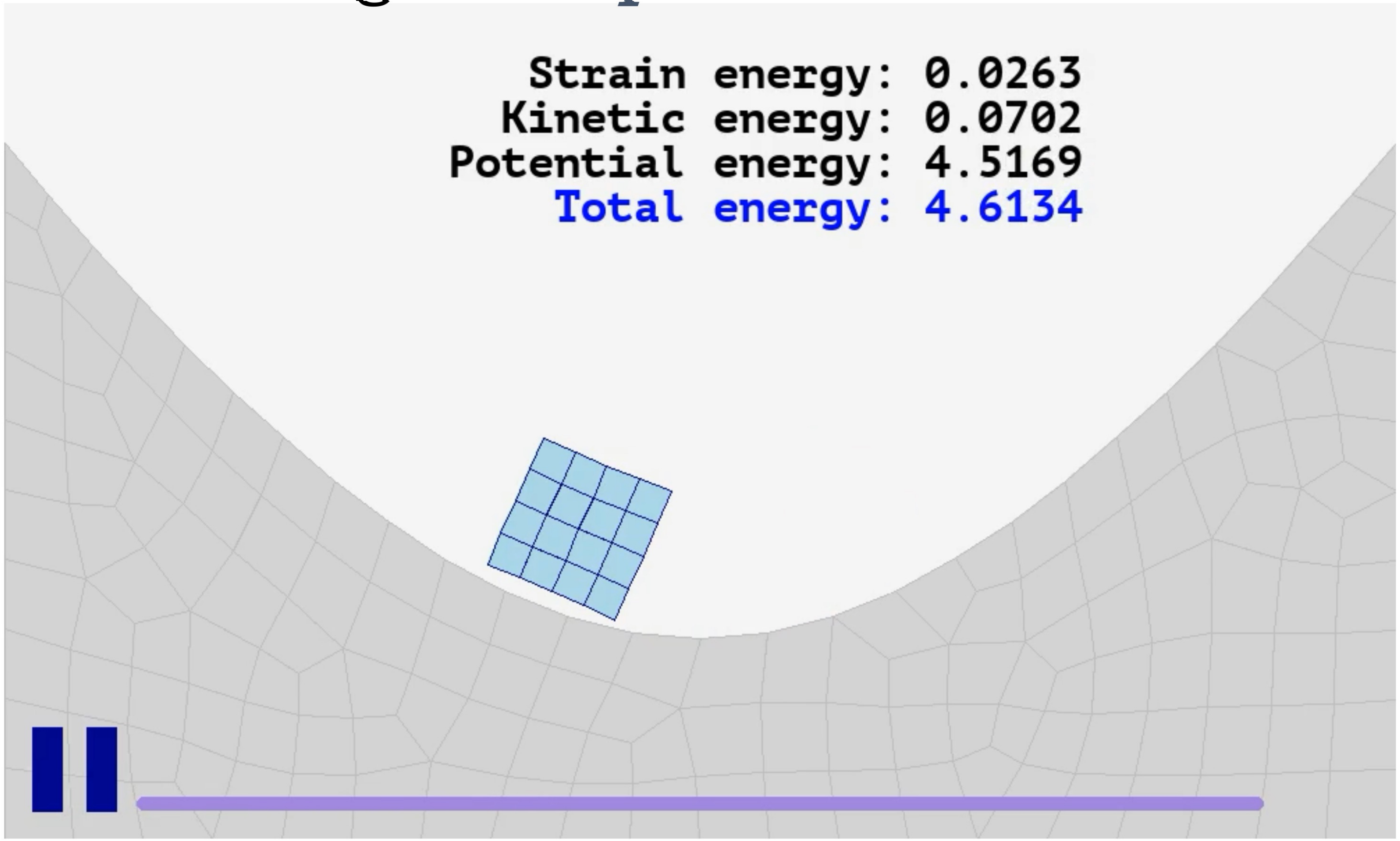


Idea previously applied to:

- **Molecular dynamics** (Levesque and Verlet, 1993)
- **Continuum mechanics** (Kum and Hoover, 1994)
- **N-body simulations** (Rein and Tamayo, 2017)
- **Chaotic dynamic systems** (Jos Stam, 2022) josstam.com/reversible

For *dissipative* dynamic systems (with damping), fixed precision arithmetic alone is insufficient to ensure exact bit-reversibility

Using *fixed-point arithmetic*:



$$\boldsymbol{M}\ddot{u} + \boldsymbol{C}\dot{u} + \boldsymbol{K}u = \boldsymbol{F}$$

damping

$$v_n \leftarrow v_{n-½} + a(u_n)\Delta t/2$$

$$v_n \leftarrow v_n \times \frac{M - C\Delta t/2}{M + C\Delta t/2}$$

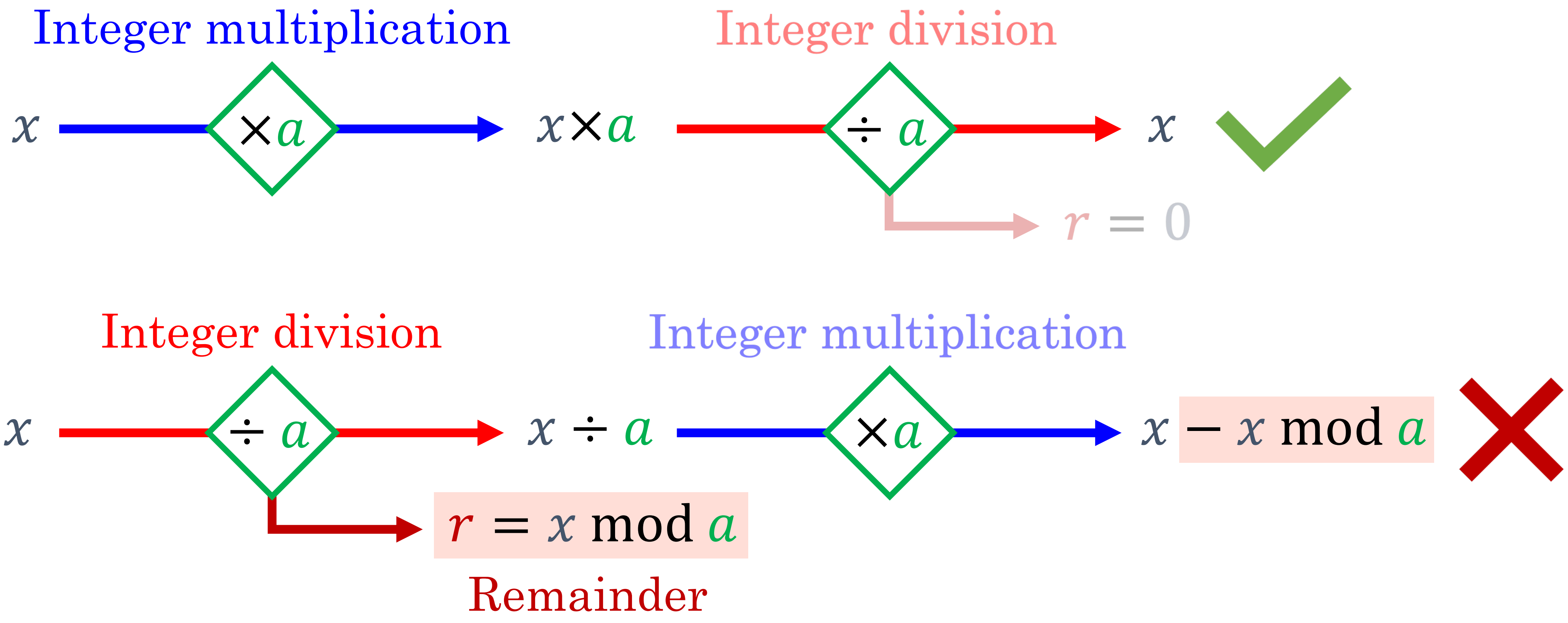
$$v_{n+½} \leftarrow v_n + a(u_n)\Delta t/2$$

$$u_{n+1} \leftarrow u_n + v_{n+½}\Delta t$$

Addition/subtraction: *reversible*

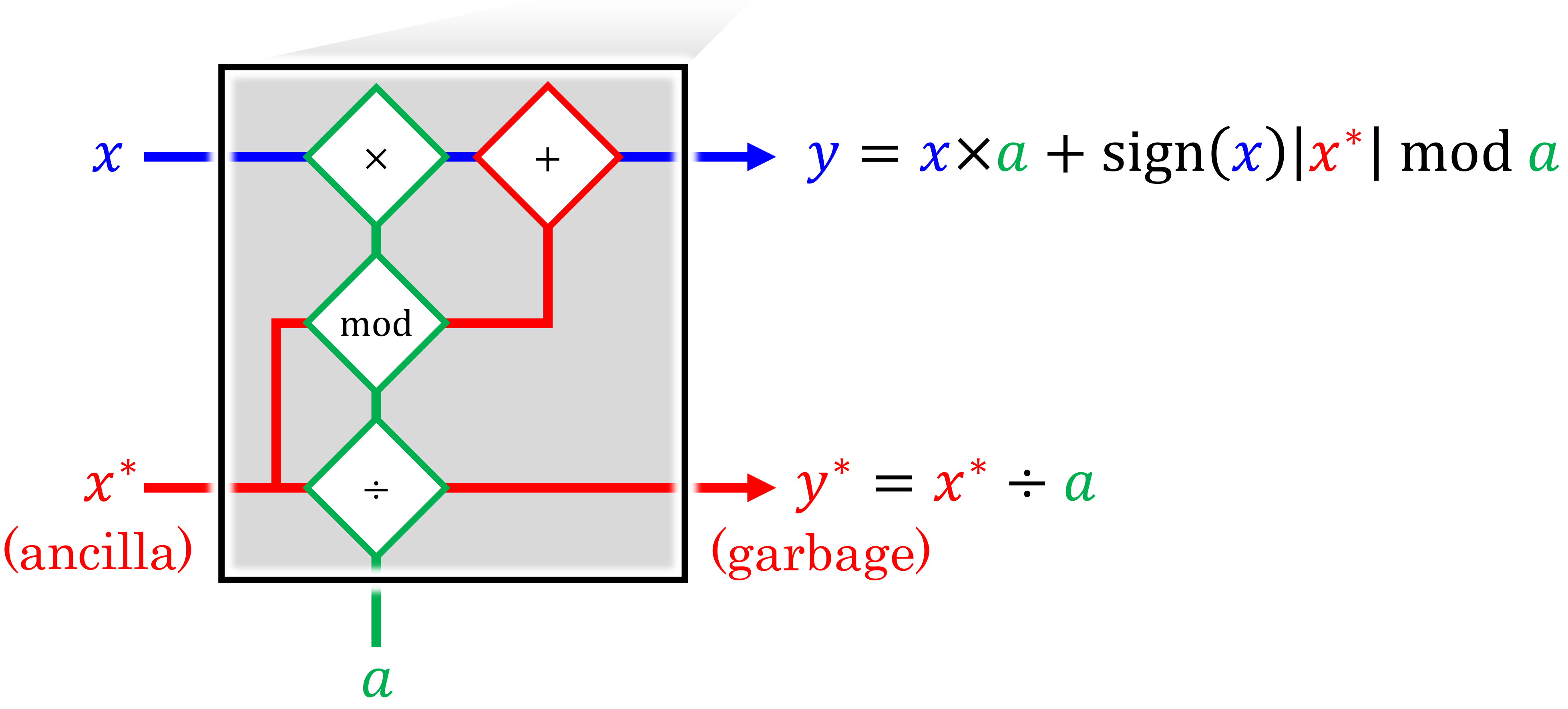
Multiplication/division: *not reversible*

Euclidean division of integers results in permanent loss (“dissipation”) of information in the form of the *remainder* after division



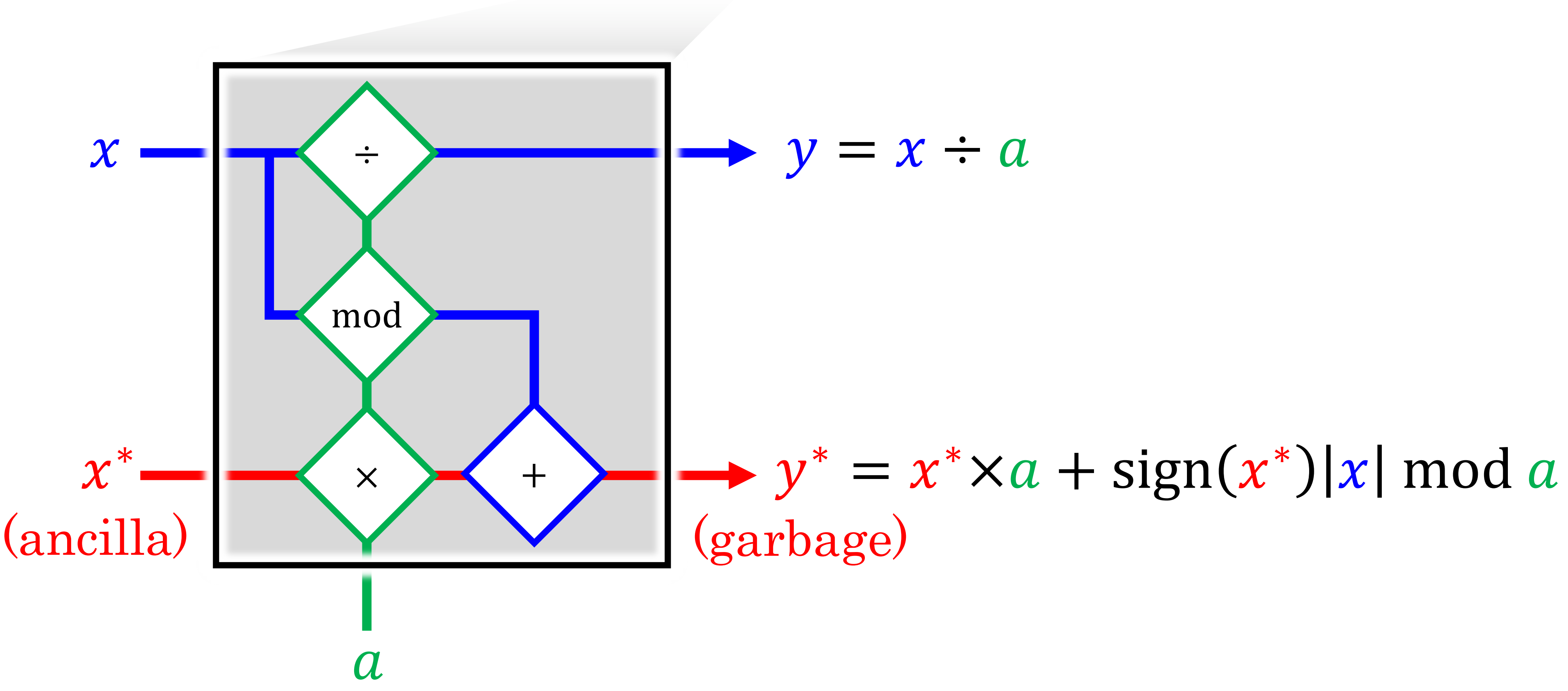
Use the round-off bit buffering approach proposed by Maclaurin et al. (2015) to define a *paired integer multiplication/division operation*

$$(x, x^*)[\times, \div]a = (y, y^*)$$



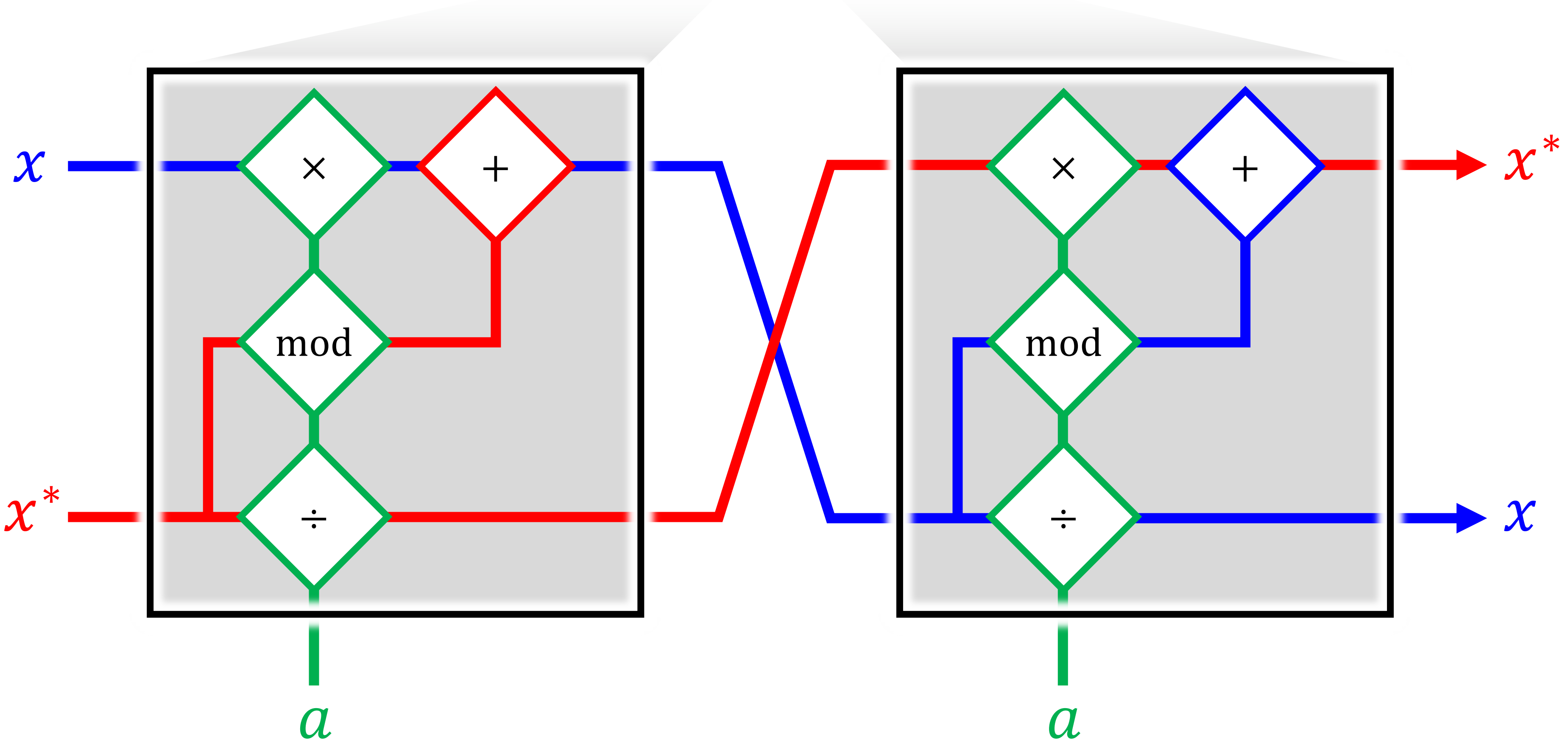
The inverse paired division/multiplication operation is obtained by simply permuting the inputs

$$(x, x^*)[\div,\times]a = (y, y^*)$$



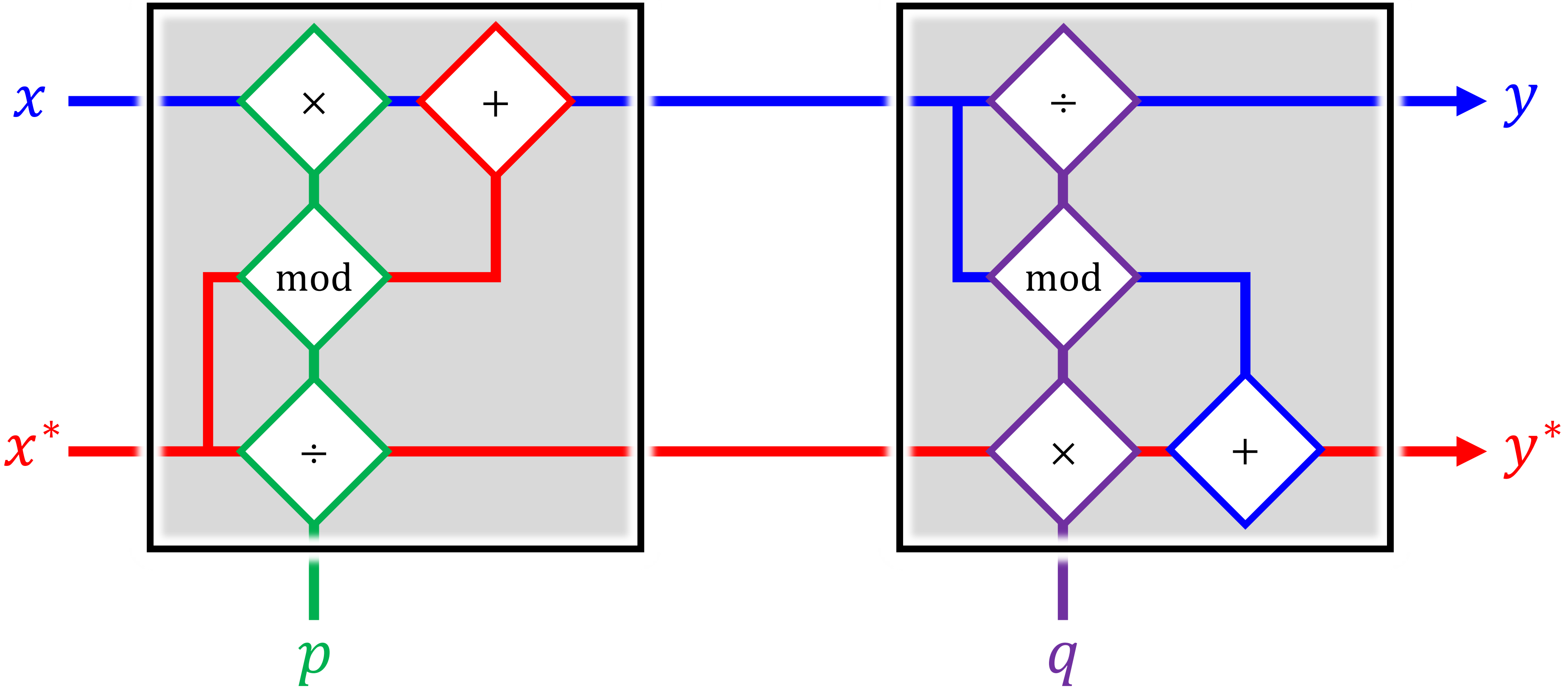
The paired integer multiplication/division operation and its permuted inverse are *exactly bit-reversible*

$$\big((x, x^*)[\times,\div]a\big)[\div,\times]a = (x, x^*)$$



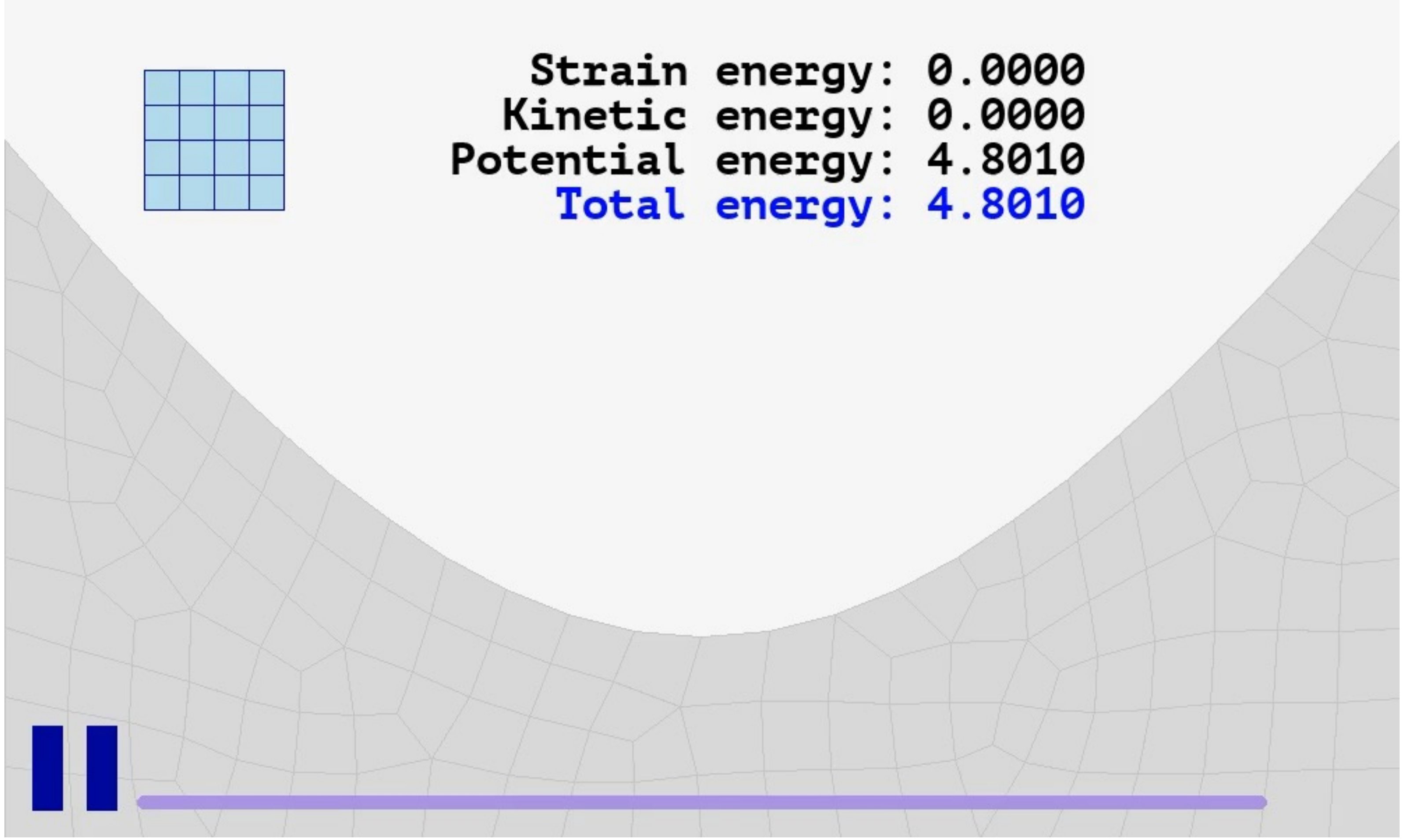
Bit-reversible fixed-point multiplication is carried out by approximating the multiplicand as a *rational number*

$$(x, x^*)[\times,\div]\frac{p}{q} = ((x, x^*)[\times,\div]p)[\div,\times]q = (y, y^*)$$



Using *paired integer multiplication/division* ensures bit-reversibility for dissipative dynamic systems

Using *paired integer multiplication/division*:



$$M\ddot{u} + C\dot{u} + Ku = F$$

mass-proportional damping:

$$C = \alpha M$$

$$v_n \leftarrow v_{n-½} + a(u_n)\Delta t/2$$

$$(v_n, v_{n+1}^*) \leftarrow (v_n, v_n^*)[\times,\div]\frac{1-\alpha\Delta t/2}{1+\alpha\Delta t/2}$$

$$v_{n+½} \leftarrow v_n + a(u_n)\Delta t/2$$

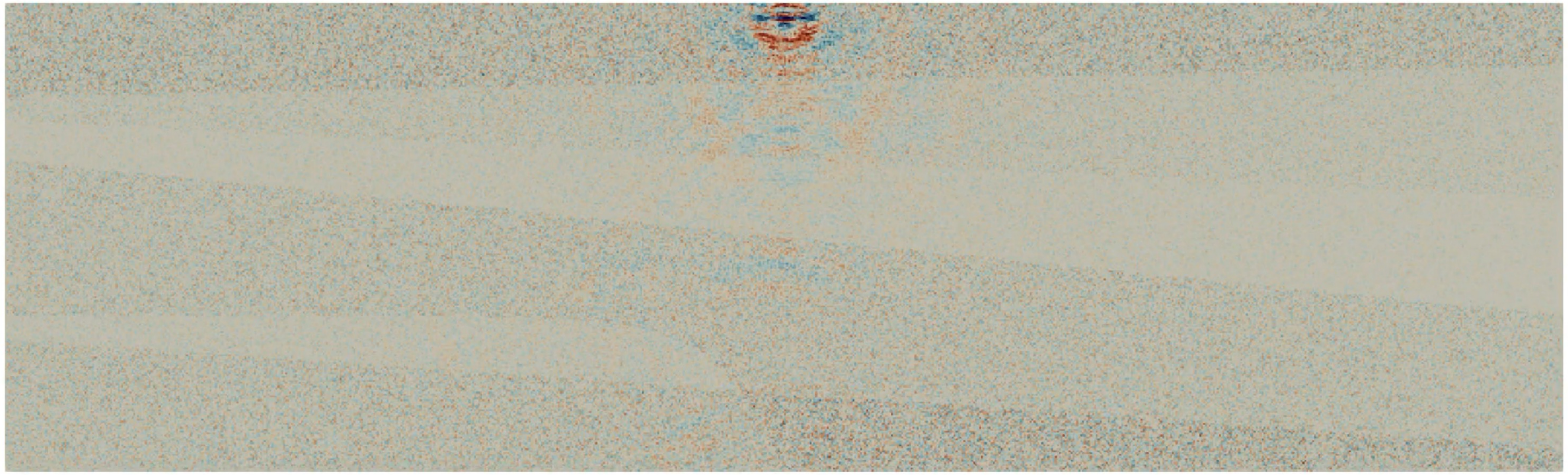
$$u_{n+1} \leftarrow u_n + v_{n+½}\Delta t$$

Ancilla velocity state variables: v_n^*

Addition/subtraction: *reversible*

Paired multiplication/division: *reversible*

Using *paired integer multiplication/division* ensures bit-reversibility for dissipative dynamic systems



Fixed-point arithmetic

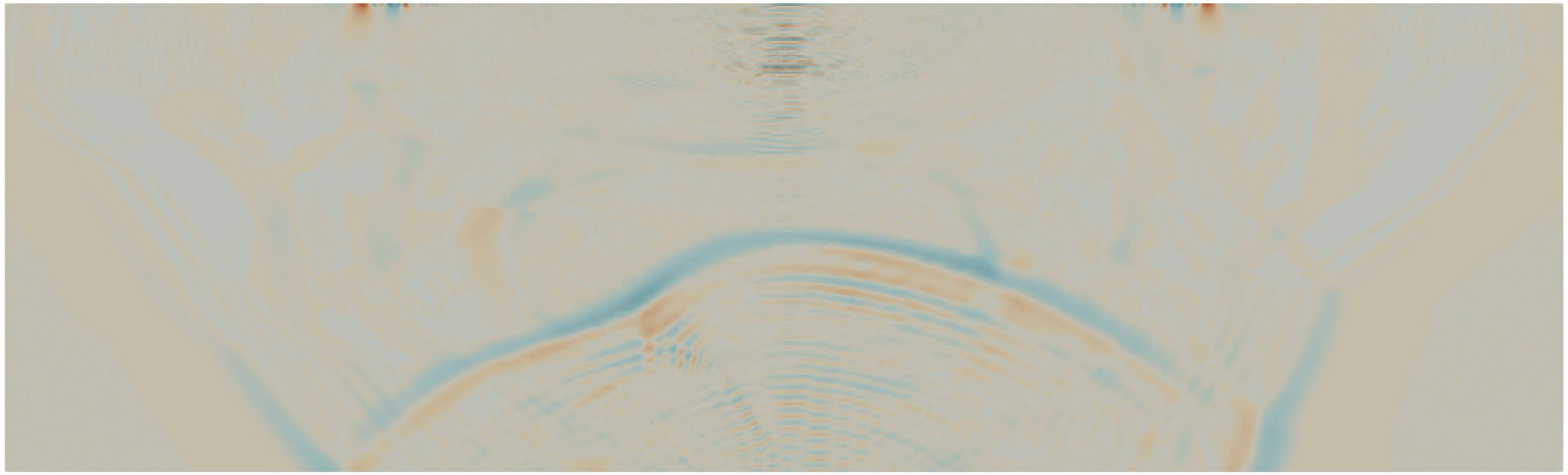
unpaired integer multiplication/division (*not* reversible)

Fixed-point arithmetic

paired integer multiplication/division (reversible)

386k degrees of freedom 1000 time steps

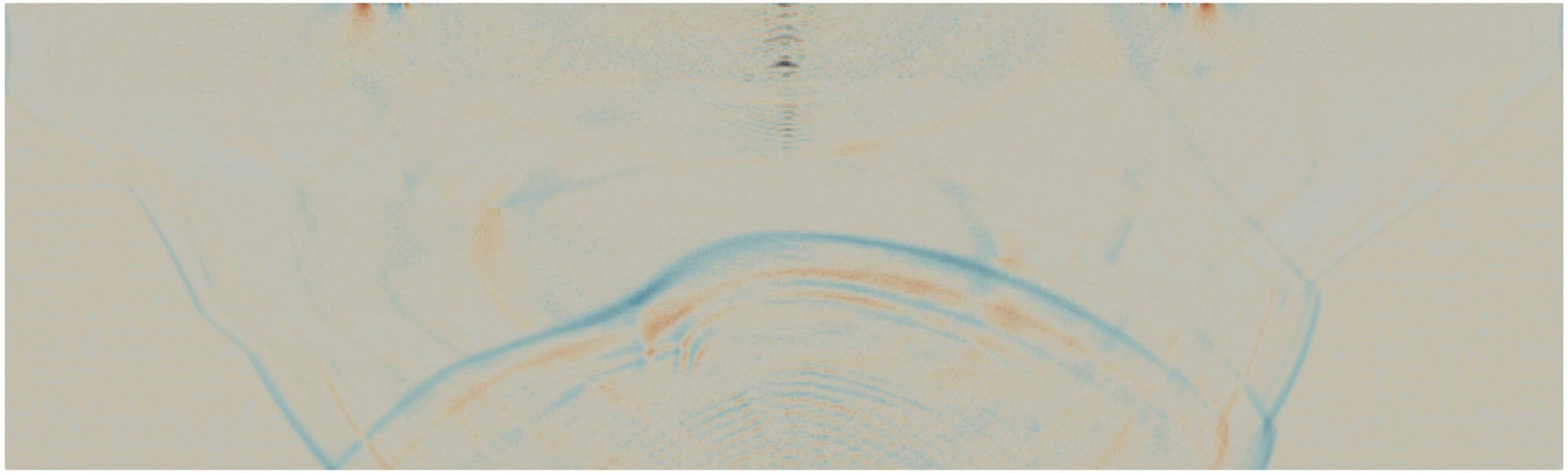
Forward solution accuracy, memory, and run-time performance using paired fixed-point arithmetic is comparable to that of floating-point arithmetic



Floating-point

1×{64-bit float (double)}

Run-time: 47.3 s



Fixed-point (reversible)

2×{32-bit int}

Run-time: 50.2 s

(6% slower)

386k degrees of freedom 1000 time steps

The concept of bit-reversible scalar multiplication/division can be generalized to achieve bit-reversible *matrix multiplication/inversion*

$$\boldsymbol{A} = \boldsymbol{LDU}$$

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ L_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & 1 \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} D_{11} & 0 & \cdots & 0 \\ 0 & D_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{NN} \end{bmatrix} \quad \boldsymbol{U} = \begin{bmatrix} 1 & U_{12} & \cdots & U_{1N} \\ 0 & 1 & \cdots & U_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$D_{ii} \approx \frac{b_i}{d_i} \in \mathbb{Q}$$

$\forall i = 1, \ldots, N$ $\qquad\qquad$ $\forall i = N, \ldots, 1$

$$\boxed{y_i = \hat{y}_i + \textstyle\sum_{j<i} L_{ij}\hat{y}_j} \Leftarrow \boxed{(\hat{y}_i, \hat{y}_i^*) = (\hat{x}_i, \hat{x}_i^*)[\times, \div]\frac{b_i}{d_i}} \Leftarrow \boxed{\hat{x}_i = x_i + \textstyle\sum_{j>i} U_{ij} x_j}$$

$$\underbrace{\boldsymbol{y} = \boldsymbol{L}\hat{\boldsymbol{y}} \qquad \hat{\boldsymbol{y}} = \boldsymbol{D}\hat{\boldsymbol{x}} \qquad \hat{\boldsymbol{y}}^* = \hat{\boldsymbol{x}}^*\boldsymbol{D}^{-1} \qquad \hat{\boldsymbol{x}} = \boldsymbol{U}\boldsymbol{x}}_{\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}}$$

The concept of bit-reversible scalar multiplication/division can be generalized to achieve bit-reversible *matrix multiplication/inversion*

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{U}$$

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ L_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & 1 \end{bmatrix} \quad \boldsymbol{D} = \begin{bmatrix} D_{11} & 0 & \cdots & 0 \\ 0 & D_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{NN} \end{bmatrix} \quad \boldsymbol{U} = \begin{bmatrix} 1 & U_{12} & \cdots & U_{1N} \\ 0 & 1 & \cdots & U_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$D_{ii} \approx \frac{b_i}{d_i} \in \mathbb{Q}$$

$\forall i = 1, \ldots, N$ $\qquad$ $\forall i = N, \ldots, 1$

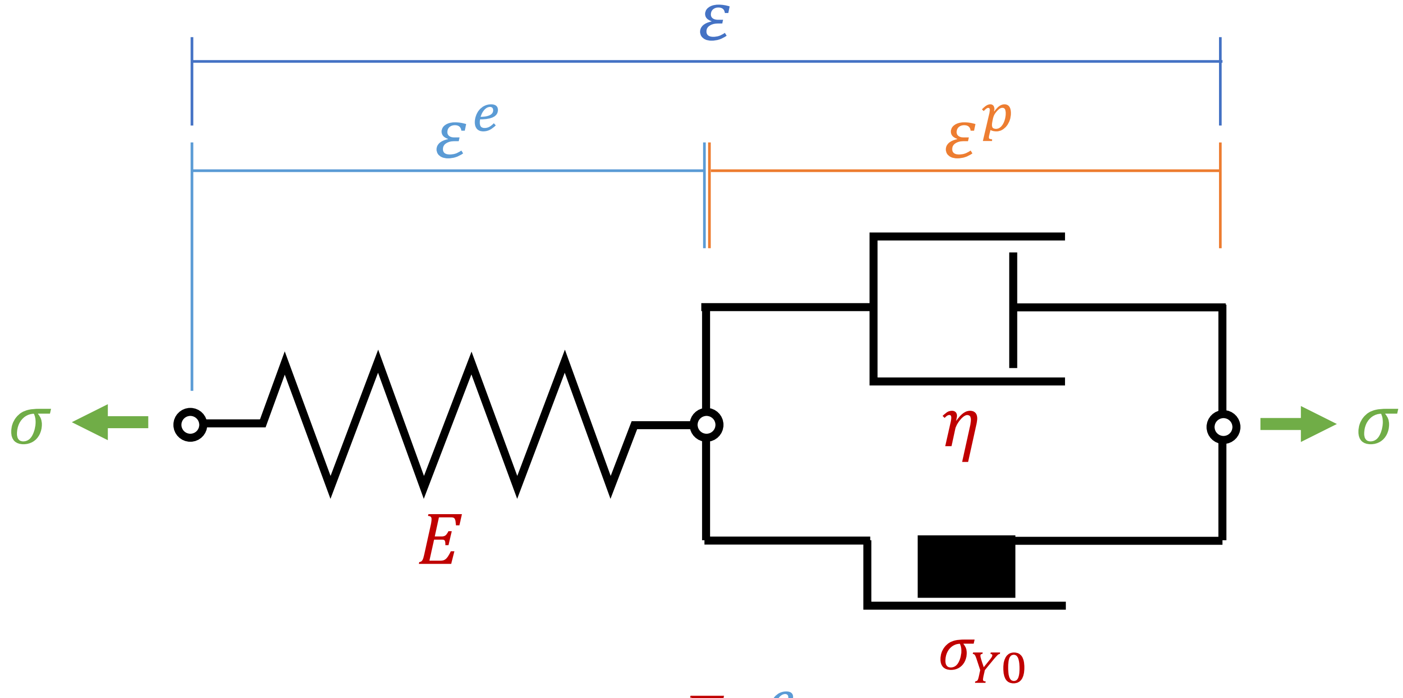
$$\boxed{\hat{y}_i = y_i - \sum_{j<i} L_{ij}\hat{y}_j} \Rightarrow \boxed{(\hat{x}_i, \hat{x}_i^*) = (\hat{y}_i, \hat{y}_i^*)[\div,\times]\frac{b_i}{d_i}} \Rightarrow \boxed{x_i = \hat{x}_i - \sum_{j>i} U_{ij}x_j}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{L}^{-1}\boldsymbol{y} \qquad \hat{\boldsymbol{x}} = \boldsymbol{D}^{-1}\hat{\boldsymbol{y}} \qquad \hat{\boldsymbol{x}}^* = \hat{\boldsymbol{y}}^*\boldsymbol{D} \qquad \boldsymbol{x} = \boldsymbol{U}^{-1}\hat{\boldsymbol{x}}$$

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$$

The proposed set of bit-reversible operations can be used to implement reversible time-integrators for common visco-elastic/plastic constitutive models

Bingham-Maxwell Model:



$$\sigma = E\varepsilon^e$$

$$|\sigma - \eta\dot{\varepsilon}^p| \leq \sigma_{Y0}$$

$$\lambda = e^{-E\Delta t/\eta}$$

$$\varepsilon_0^p = \varepsilon - \mathrm{sign}(\varepsilon - \varepsilon^p)\sigma_{Y0}/E$$

Forward:

If $|E(\varepsilon - \varepsilon^p)| > \sigma_{Y0}$

$$(\varepsilon^p, \varepsilon^{p*}) \leftarrow (\varepsilon^p, \varepsilon^{p*})[\times, \div]\lambda$$

$$\varepsilon^p \leftarrow \varepsilon^p + \varepsilon_0^p(1 - \lambda)$$

Backward:

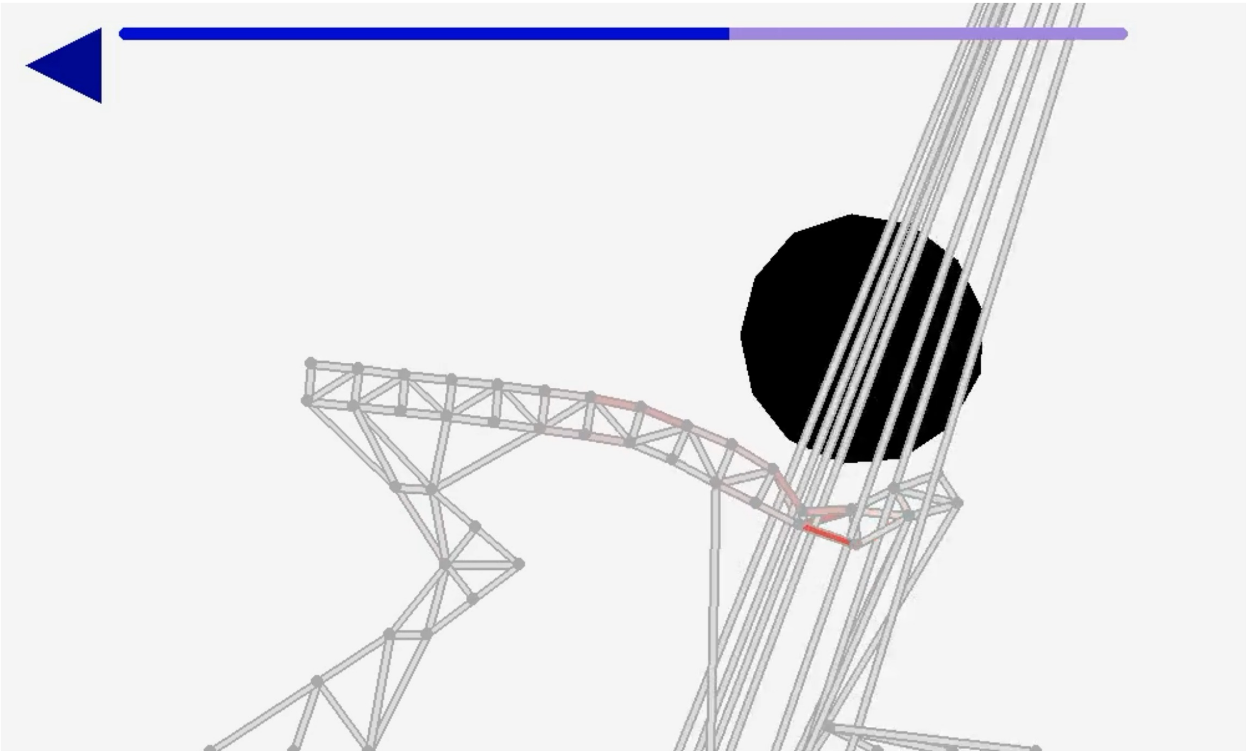
If $|E(\varepsilon - \varepsilon^p)| > \sigma_{Y0}$

$$\varepsilon^p \leftarrow \varepsilon^p - \varepsilon_0^p(1 - \lambda)$$

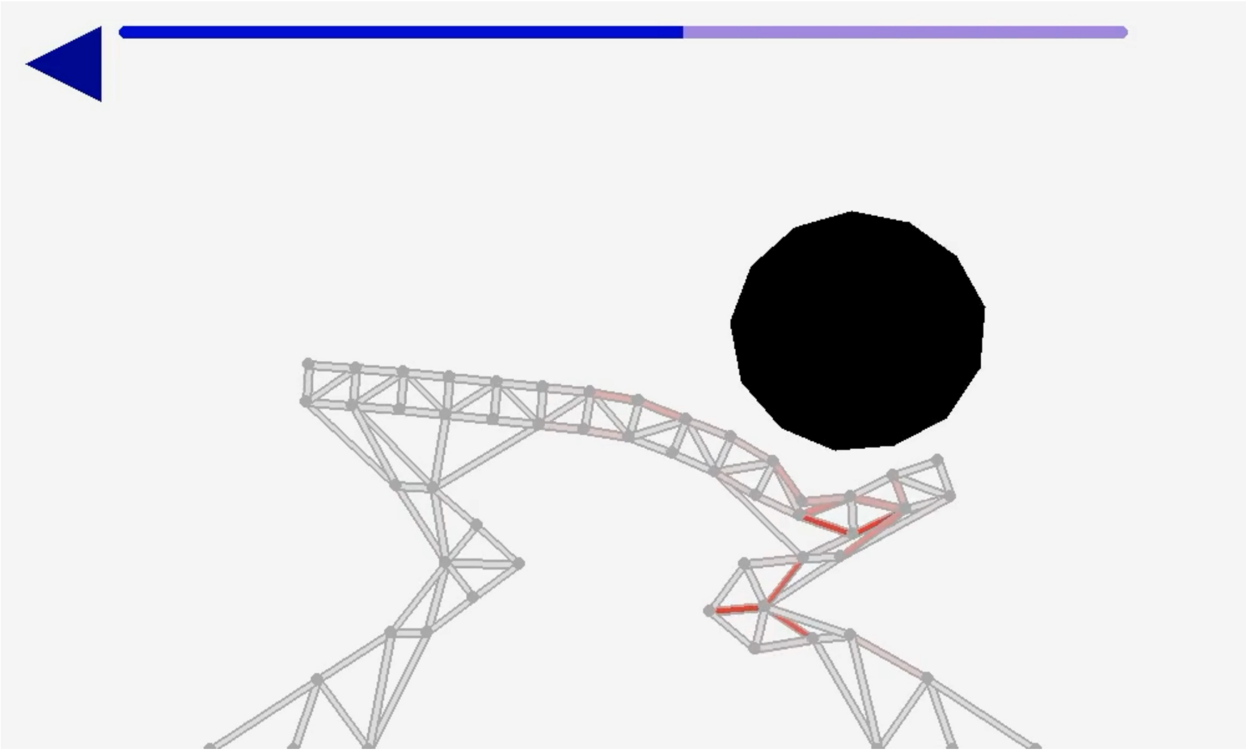
$$(\varepsilon^p, \varepsilon^{p*}) \leftarrow (\varepsilon^p, \varepsilon^{p*})[\div, \times]\lambda$$

Demonstration: implementation of a reversible uniaxial visco-plasticity model with a maximum plastic strain-based failure criterion

Floating-point (*not* reversible)



Fixed-point (reversible)



Ongoing and future work

- Limitations:
    - **Overflow**!
    - Not all models are amenable to a reversible implementation
    - Must ensure *consistent* execution during forward/backward passes
- Alternative bit-roundoff data compression methods
- Inelastic material behavior
    - Continuum damage/plasticity/visco-elasticity, fracture, friction
- Compare performance on GPUs
    - Does the proposed approach help with minimizing device I/O and latency?
- Application to optimization/inverse problems
    - Optimal design of impact-resistant structures

Questions?