Lecture 3

Particle Systems & ODEs

FUNDAMENTALS OF COMPUTER GRAPHICS

Animation & Simulation

Stanford CS248B, Fall 2023

PROFS. KAREN LIU & DOUG JAMES

Announcements

- HW1 out on Canvas.
  - Due next Tuesday by midnight
  - Submit on Gradescope (Q1,3,4) & OpenProcessing (Q2)
  - Last question is related to material from this class (ODE integrators)
- Grade breakdown for course:

Material	Weight
HW1	10%
HW2	10%
HW3	10%
HW4	10%
P0	0%
P1 (Pinball)	20%
P2 (Blobs)	20%
P3 (IK)	20%

Overview

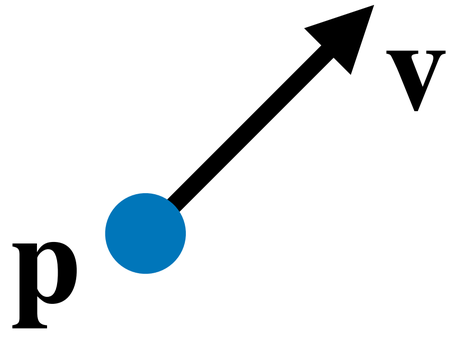
- Particle Systems
- Ordinary Differential Equations (ODEs)
  - Reduction to first-order form. Autonomous vs nonautonomous. Linearization. Model equation. Stability.
  - Time-stepping schemes (forward Euler, backward Euler, symplectic Euler, midpoint, ...).
  - Stiffness and stability.

Particle Systems

One lousy particle

- Consider a single particle in n dimensions
- Kinematic model of particle motion (vs *dynamics* which involves forces)
- Position, $\mathbf{p} = \mathbf{p}(t) \in \mathbb{R}^n$
- Velocity, $\mathbf{v} = \mathbf{v}(t) \in \mathbb{R}^n$
- Differential relationship

$$\mathbf{v} = \frac{d\mathbf{p}}{dt} = \dot{\mathbf{p}}$$



- Example: 2D ball

$$\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \quad \text{and} \quad \mathbf{v}(t) = \begin{pmatrix} v_x(t) \\ v_y(t) \end{pmatrix} = \begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix}$$

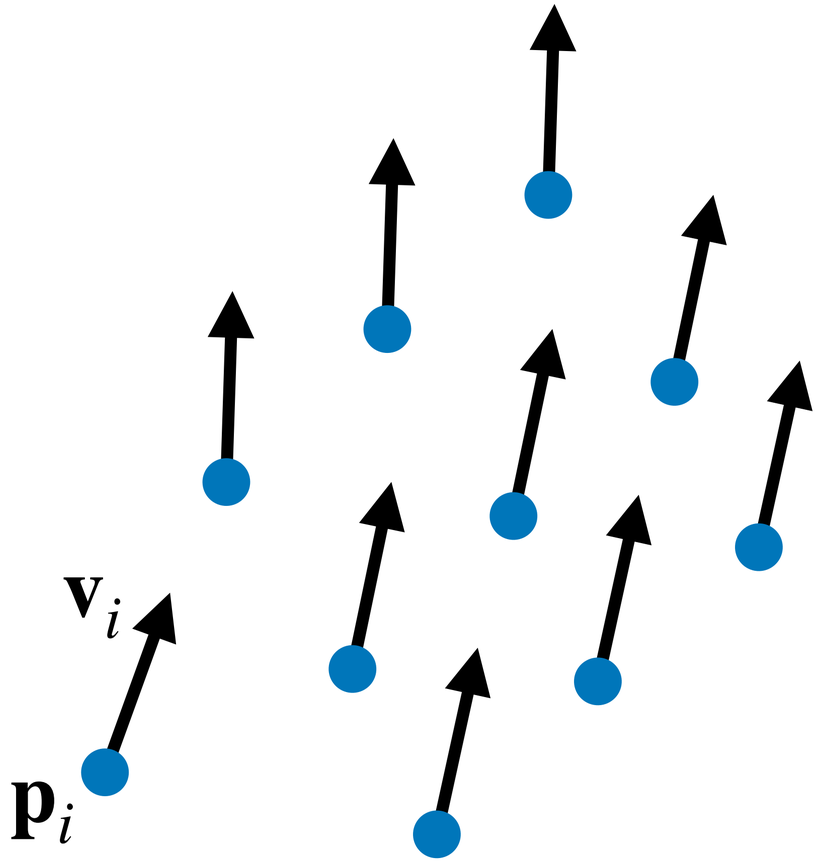
Particle systems

- Collection of N particles in $\mathbb{R}^n$
  - Created or destroyed by various means.
- Indexed particle values:

$$\mathbf{p}_i = \mathbf{p}_i(t) \in \mathbb{R}^n, \quad \text{for} \quad i = 1,\ldots,N.$$
$$\mathbf{v}_i = \mathbf{v}_i(t) \in \mathbb{R}^n, \quad \text{for} \quad i = 1,\ldots,N.$$

- Vector form: $\mathbf{v} = \dot{\mathbf{p}} \in \mathbb{R}^{nN}$ where

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_i \\ \vdots \\ \mathbf{p}_N \end{pmatrix} \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_i \\ \vdots \\ \mathbf{v}_N \end{pmatrix}$$

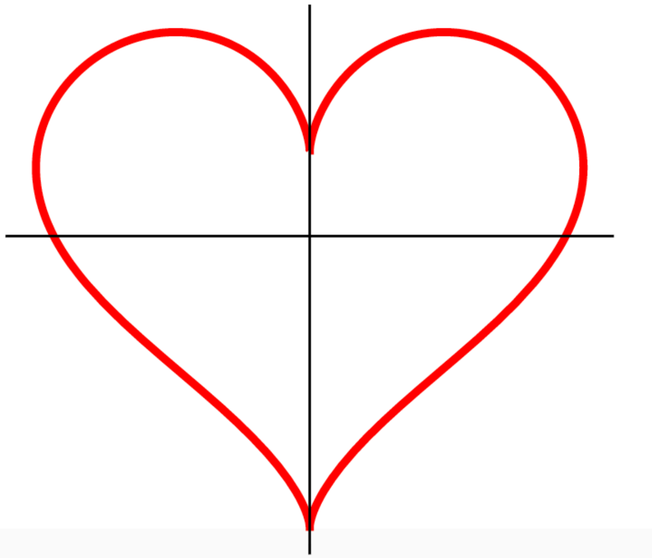


Zeroth-order dynamics

- Directly specify position, $\mathbf{p}(t)$, function
  - Zeroth position derivative
- Examples:
  - Parametric equations, e.g., heart curve
  - Keyframe animation (more later)
- Pros:
  - Full control
  - No simulation or dynamics
- Cons:
  - Tedious & hard to do complex motions
  - No physics, collisions or other interactions

$$x = 16\sin^3(t)$$

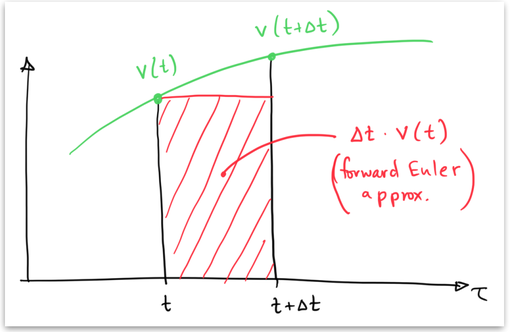
$$y = 13\cos(t) - 5\cos(2t) - 2\cos(3t) - \cos(4t)$$



https://mathworld.wolfram.com/HeartCurve.html
https://www.openprocessing.org/sketch/966495

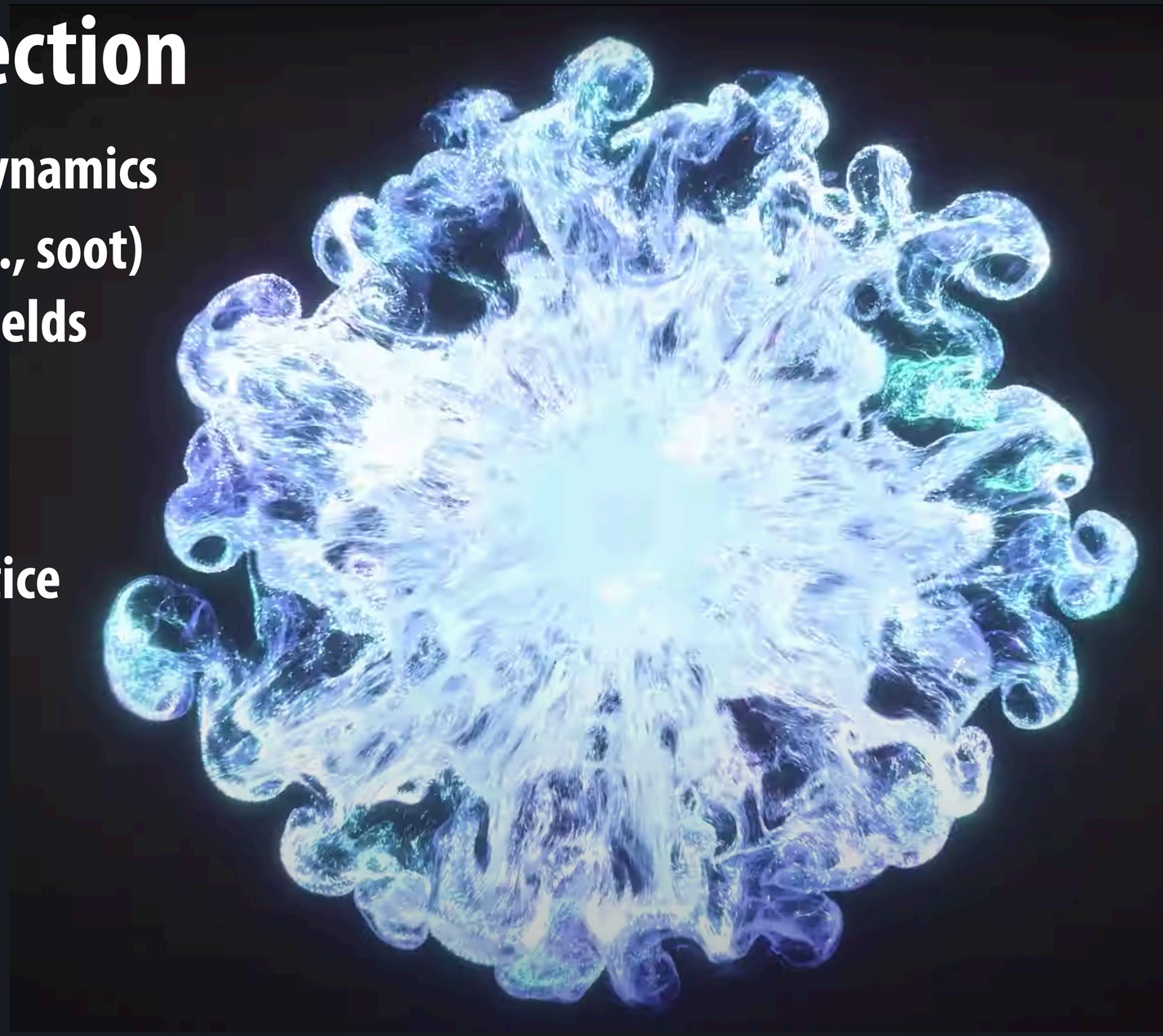
First-order dynamics

- Specify velocity, $\mathbf{v}(t)$
  - Integrate $\mathbf{v}(t)$ to get $\mathbf{p}(t)$ of particles
  - Solve an ordinary differential equation (ODE)
- Solve an *initial value problem* (IVP):
  - Solve $\dot{\mathbf{p}} = \mathbf{v}$ for $\mathbf{p}(t)$ such that $\mathbf{p}(0) = \bar{\mathbf{p}}$.
  - In general, input velocity is given by some gradient function $\mathbf{v}(t) = \mathbf{V}(\mathbf{p}(t), t)$
  - Solution: $\mathbf{p}(t + \Delta t) = \mathbf{p}(t) + \int_t^{t+\Delta t} \mathbf{v}(\tau)\, d\tau$
- Numerical approximation: Forward Euler scheme
  - $\mathbf{p}(t + \Delta t) = \mathbf{p}(t) + \mathbf{v}(t)\, \Delta t$
  - $O(\Delta t^2)$ truncation error



Application: Particle advection

- Important application of first-order dynamics
  - Advection of massless particles (e.g., soot) in known spatiotemporal velocity fields
- Integrate $\dot{\mathbf{p}} = \mathbf{V}(\mathbf{p}, t)$
  - subject to initial condition, $\mathbf{p}(0) = \bar{\mathbf{p}}$.
- Millions to billions of particles in practice
- Pleasantly parallel integration



https://www.youtube.com/watch?v=f51ScQ162wA&ab_channel=HAGI

Application: Energy minimization

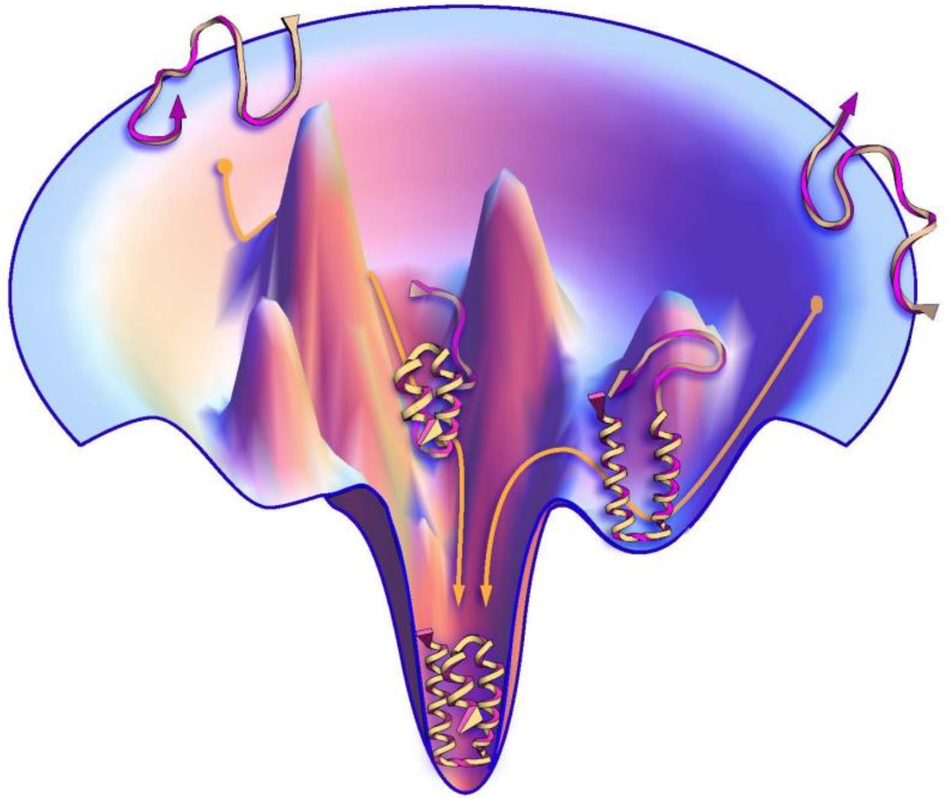
- Consider particular motion under gradient descent

$$\min_{\mathbf{p}} E(\mathbf{p})$$

- Numerical approximation

$$\mathbf{p}_{i+1} = \mathbf{p}_i - h \nabla E(\mathbf{p}_i)$$

$$\xrightarrow{h \to 0} \quad \frac{d\mathbf{p}}{dt} = -\nabla E(\mathbf{p})$$



https://www.sciencedaily.com/releases/2012/11/121122152910.htm

Application: Sampling Implicit Surfaces

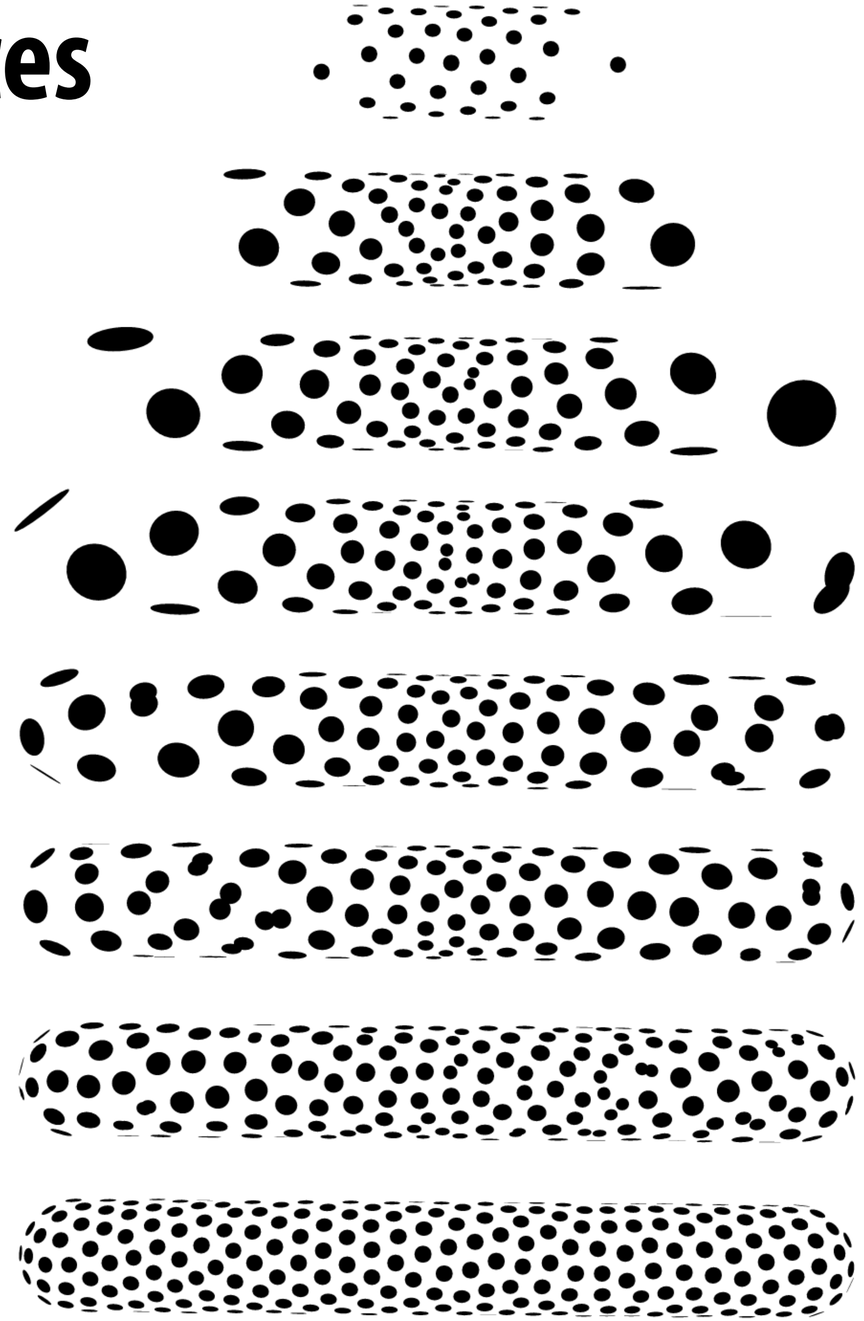
- Witkin and Heckbert, "Using Particles to Sample and Control Implicit Surfaces," SIGGRAPH 94.
- Particles
  - Constrained to lie on an iso-surface, $d(\mathbf{p}) = 0$
  - Repel each other using the energy:

$$E_{ij} = e^{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\sigma_i^2}}$$

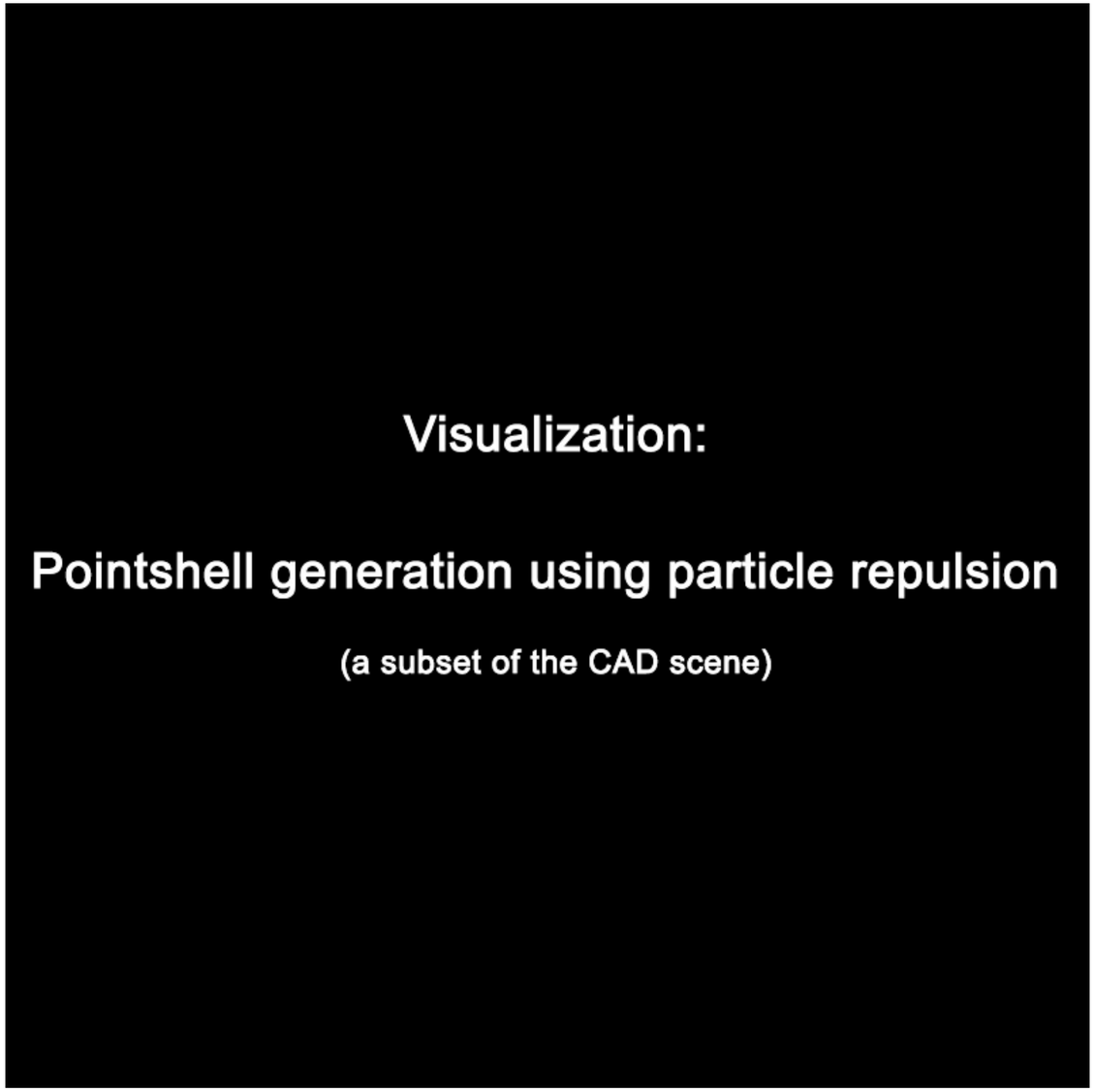
  - Particle i energy:

$$E_i(\mathbf{p}) = \sum_{j=1}^{N} \left(E_{ij} + E_{ji} \right)$$

  - Move particles around using gradient descent



Application: Sampling Implicit Surfaces



From [Barbic and James 2007]

Second-order dynamics: "f=ma"

- Specify particle acceleration, $\mathbf{a} = \dfrac{d\mathbf{p}^2}{dt^2}$, and integrate twice.
- Newton's equations of motion for a particle system:

$$m_i \frac{d\mathbf{p}_i^2}{dt^2} = \mathbf{f}_i \quad \text{for} \quad i = 1, \ldots, N\,.$$

  - m_i is the mass of particle i
  - $\mathbf{f}_i = \mathbf{f}_i(\mathbf{p}, \mathbf{v}, t)$ are external forces
  - Second-order ODE for $\mathbf{p}(t)$
- Matrix form
  - $\mathbf{M}\,\ddot{\mathbf{p}} = \mathbf{f}$ where $\mathbf{M} = \mathrm{diag}(m_1\mathbf{I}_n, \ldots, m_N\mathbf{I}_n)$

Simple particle forces

- **Constant gravity:**

$$\mathbf{f}_i = m\,\mathbf{g}$$

- **Aerodynamic drag:**

$$\mathbf{f}_i = -\,c\,\mathbf{v}_i, \quad c > 0$$

- **Zero-rest-length springs:**
  - **Pulling particle to anchor point, c:** $\mathbf{f}_i = -\,k\left(\mathbf{p}_i - \mathbf{c}\right)$
  - **Pulling two particles together:** $\mathbf{f}_{i/j} = \pm\,k\left(\mathbf{p}_i - \mathbf{p}_j\right)$
- **Newton's law of gravitation (c.f. electrostatic forces)**

$$\mathbf{f}_{i/j} = \pm\,G\,m_i\,m_j\,\frac{(\mathbf{p}_i - \mathbf{p}_j)}{\|\mathbf{p}_i - \mathbf{p}_j\|^3}$$

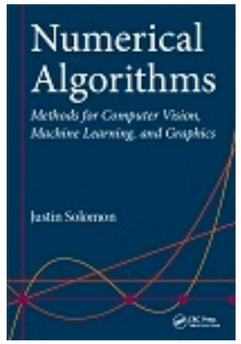
Ordinary Differential Equations (ODEs)

Basic Theory and Numerical Integration

Ordinary Differential Equations (ODEs)

- Initial value problems (IVPs)
  - Common in animation

- Textbook reference:



  - Solomon, Justin. Numerical Algorithms.
    Textbook published by AK Peters/CRC Press, 2015

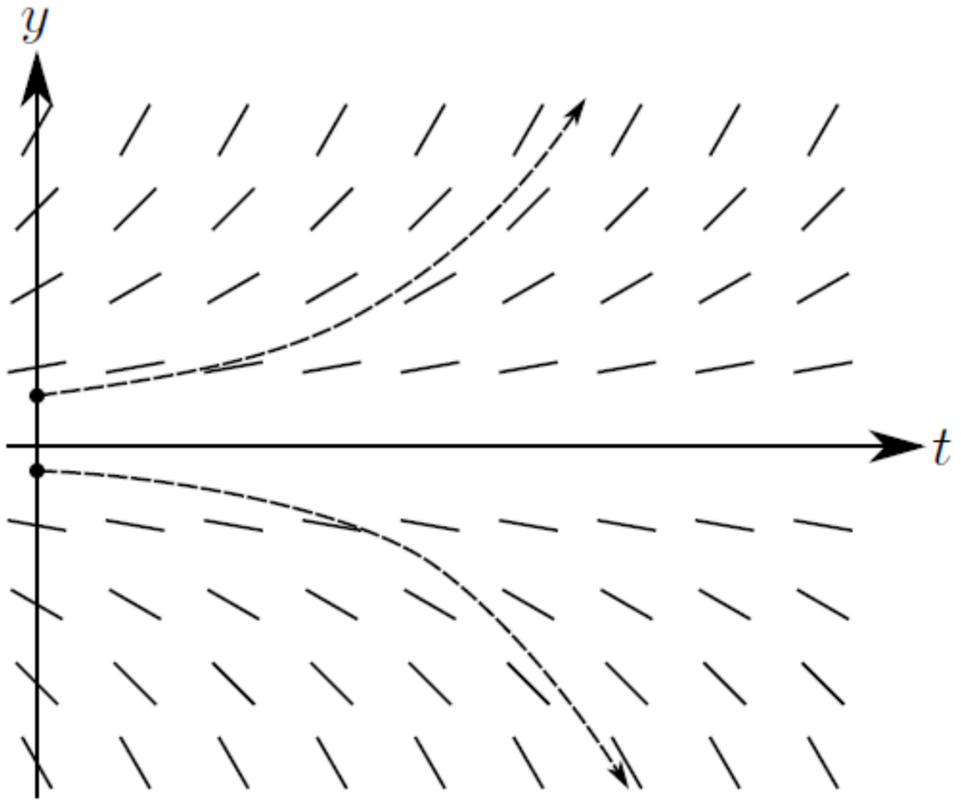
Conversion to first-order ODE form

- Can always convert a system of ODEs to first-order form
  - Useful step for applying standard ODE solvers
  - Explicit first-order form looks like $\dot{\mathbf{y}} = \mathbf{F}(\mathbf{y}, t)$
- Convert $\mathbf{M}\ddot{\mathbf{p}} = \mathbf{f}$ by writing

$$\begin{aligned} \frac{d\mathbf{p}}{dt} &= \mathbf{v} \\ \frac{d\mathbf{v}}{dt} &= \mathbf{M}^{-1}\mathbf{f} \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{M}^{-1}\mathbf{f} \end{aligned}$$

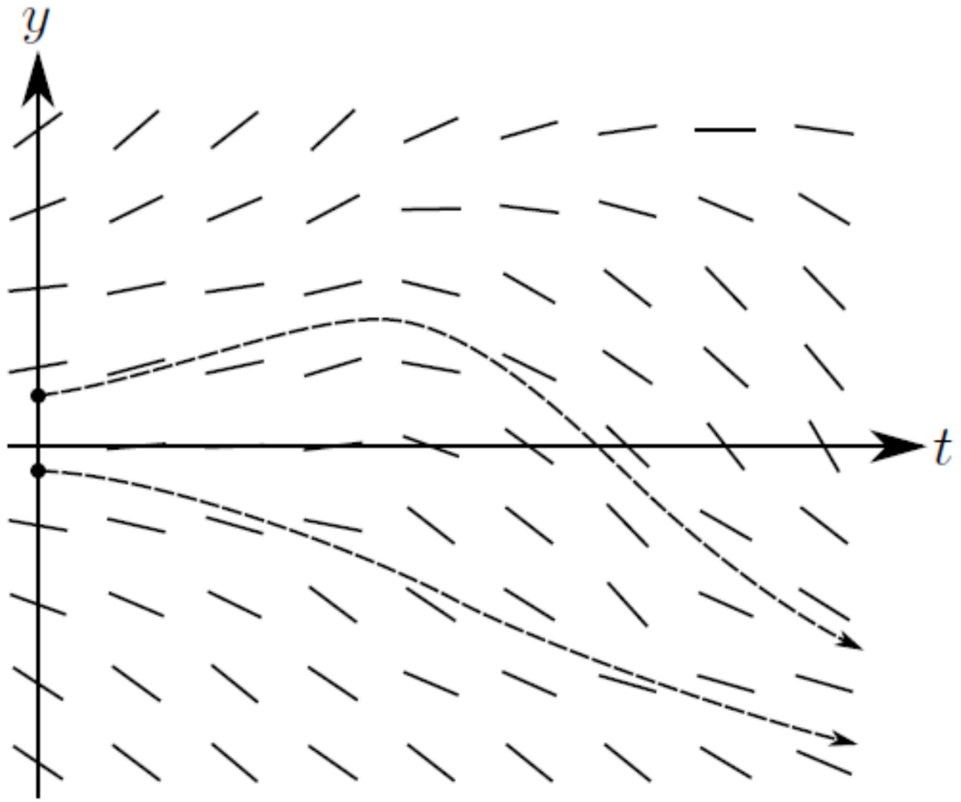
- Let $\mathbf{y} = \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \end{pmatrix} \in \mathbb{R}^{2nN}$ then $\dot{\mathbf{y}} = \mathbf{F}(\mathbf{y}, t) = \begin{pmatrix} \mathbf{v}(t) \\ \mathbf{M}^{-1}\mathbf{f}(\mathbf{y}, t) \end{pmatrix}$ s.t. $\mathbf{y}(0) = \begin{pmatrix} \bar{\mathbf{p}} \\ \bar{\mathbf{v}} \end{pmatrix}$

Time dependence of gradient function, F



Time-independent

$$\dot{y} = F(y)$$



Time-dependent

$$\dot{y} = F(y, t)$$

Autonomous & non-autonomous ODEs

- Minor issue: We have a non-autonomous ODE (explicit time dependence)

$$\dot{\mathbf{y}} = \mathbf{F}(\mathbf{y}, t)$$

but some solvers assume autonomous ODE form (no explicit time dependence)

$$\dot{\mathbf{z}} = \mathbf{G}(\mathbf{z})$$

- There's an easy fix to convert a non-autonomous ODE, $\dot{\mathbf{y}} = \mathbf{F}(\mathbf{y}, t)$, to autonomous form
- Introduce a time-like variable $\tau = \tau(t)$ with $\dot{\tau} = 1$, and $\tau(0) = 0$ (solution: $\tau = t$).
- Form an augmented system, and replace all RHS references to t with τ :

$$\frac{d}{dt}\begin{pmatrix}\mathbf{y}\\ \tau\end{pmatrix} = \begin{pmatrix}\mathbf{F}(\mathbf{y}, \tau)\\ 1\end{pmatrix} \quad \Longleftrightarrow \quad \dot{\mathbf{z}} = \mathbf{G}(\mathbf{z})$$

where $\mathbf{z}(t) = \begin{pmatrix}\mathbf{y}(t)\\ \tau(t)\end{pmatrix}$ and $\mathbf{z}(0) = \begin{pmatrix}\bar{\mathbf{y}}\\ 0\end{pmatrix}$.

Existence and Uniqueness of Solutions to $\mathbf{y}' = F(\mathbf{y})$

Theorem: Local existence and uniqueness

Suppose F is continuous and Lipschitz, that is, $\|F[\vec{y}] - F[\vec{x}]\|_2 \leq L\|\vec{y} - \vec{x}\|_2$ for some fixed $L \geq 0$. Then, the ODE $f'(t) = F[f(t)]$ admits exactly one solution for all $t \geq 0$ regardless of initial conditions.

- **Issue: Many problems in animation may not be sufficiently continuous due to non-smooth contact, e.g., instantaneous velocity changes.**

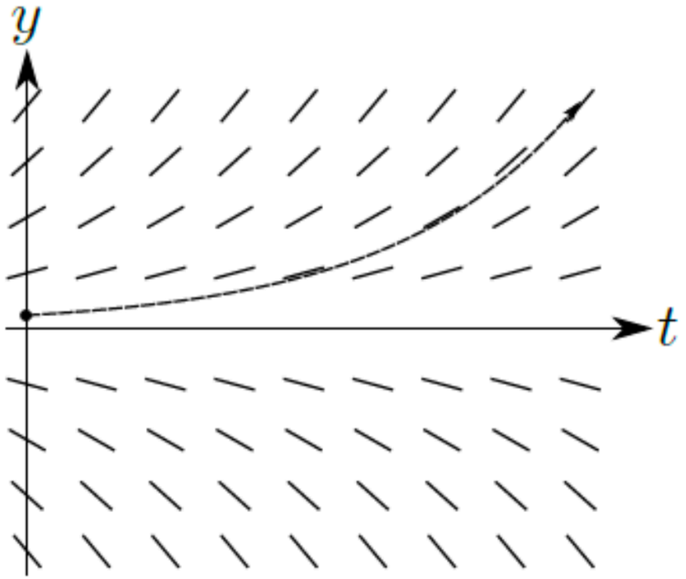
Linear Model Equations

- Common to study stability and behavior of time-stepping schemes on model equations
- Consider the 1D ODE: $\dot{y} = F(y)$
  - Linearizing problem:
    - $F(y) = F(0) + F'(0)\,y = a\,y + b = a\left(y + \frac{b}{a}\right) = a\,\bar{y}$
    - Since $\frac{d}{dt}y' = \frac{d}{dt}\bar{y}'$ it is sufficient to consider...

Model Problem:

$$\dot{y} = a\,y \implies y(t) = C\,e^{at}$$

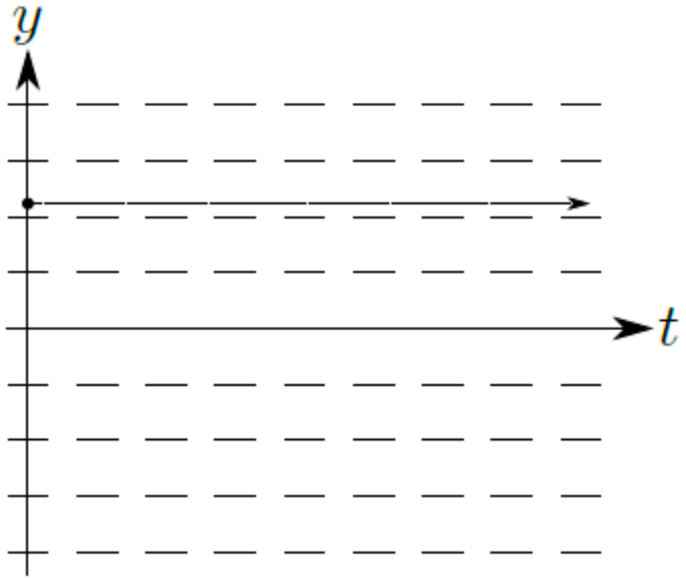
Stability: Visualization



$a > 0$

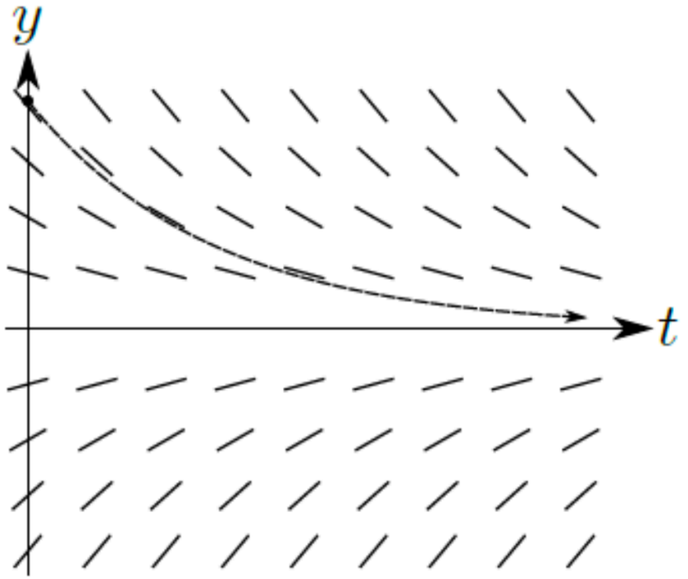
Unstable

Solutions separate



$a = 0$

Stable



$a < 0$

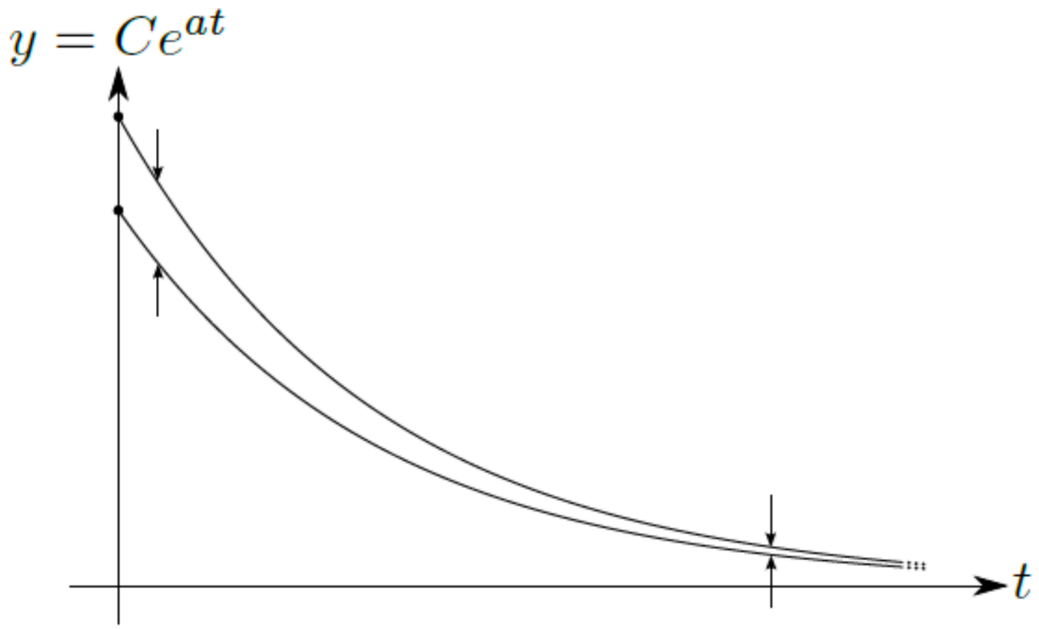
Stable

Solutions get closer

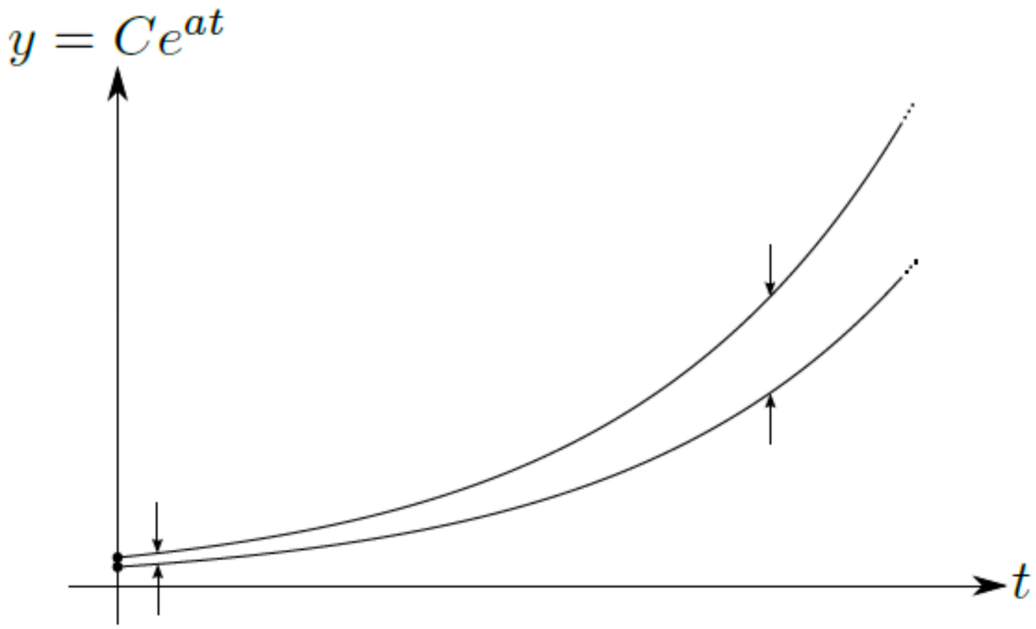
$$\dot{y} = a\,y$$

Stability and error in initial conditions

An *unstable* ODE magnifies mistakes in the initial conditions $y(0)$.



Stable ($a < 0$)



Unstable ($a > 0$)

Time-stepping schemes

- Consider $\dot{\mathbf{y}} = \mathbf{F}(\mathbf{y})$
- Given $\mathbf{y}_k$ at time t_k generate $\mathbf{y}_{k+1}$ at time $t_{k+1} = t_k + h$
- Things to consider
  - Accuracy
    - Local truncation error
    - Global truncation error
  - Stability
    - Analyze model problem $y' = ay \quad (a < 0)$
    - Stable when $|y_{k+1}| \leq |y_k|$
    - Unconditional vs conditional stability
      - Time-step restrictions

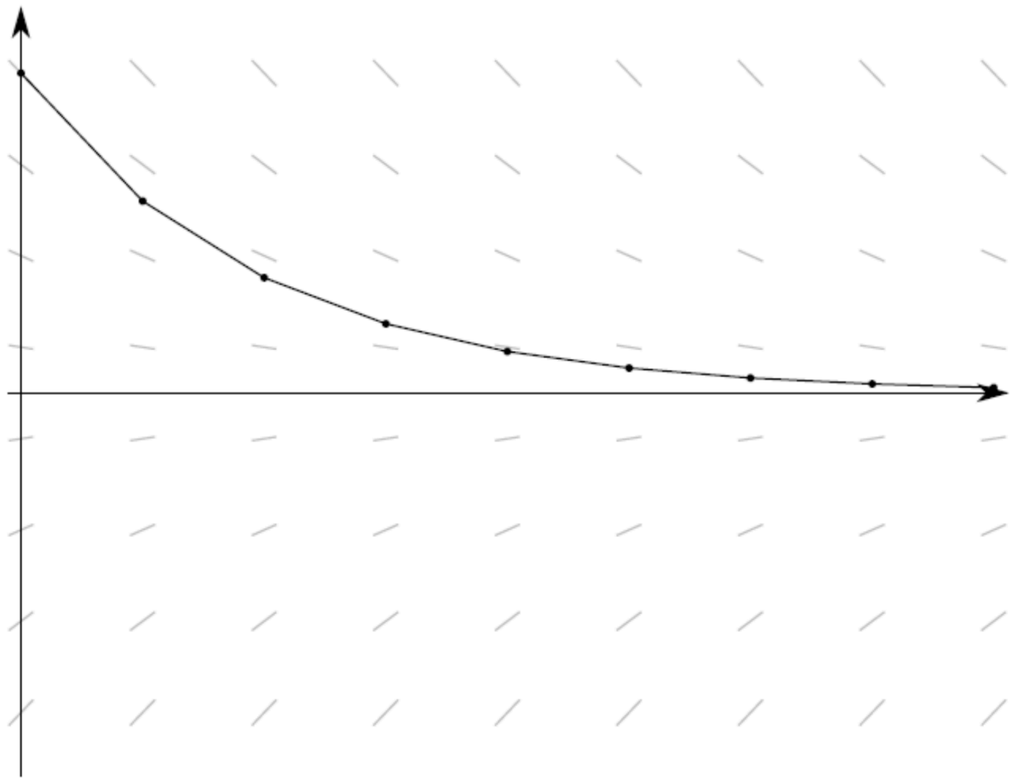
Forward Euler (a.k.a. Explicit Euler)

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\,\mathbf{F}(\mathbf{y}_k)$$

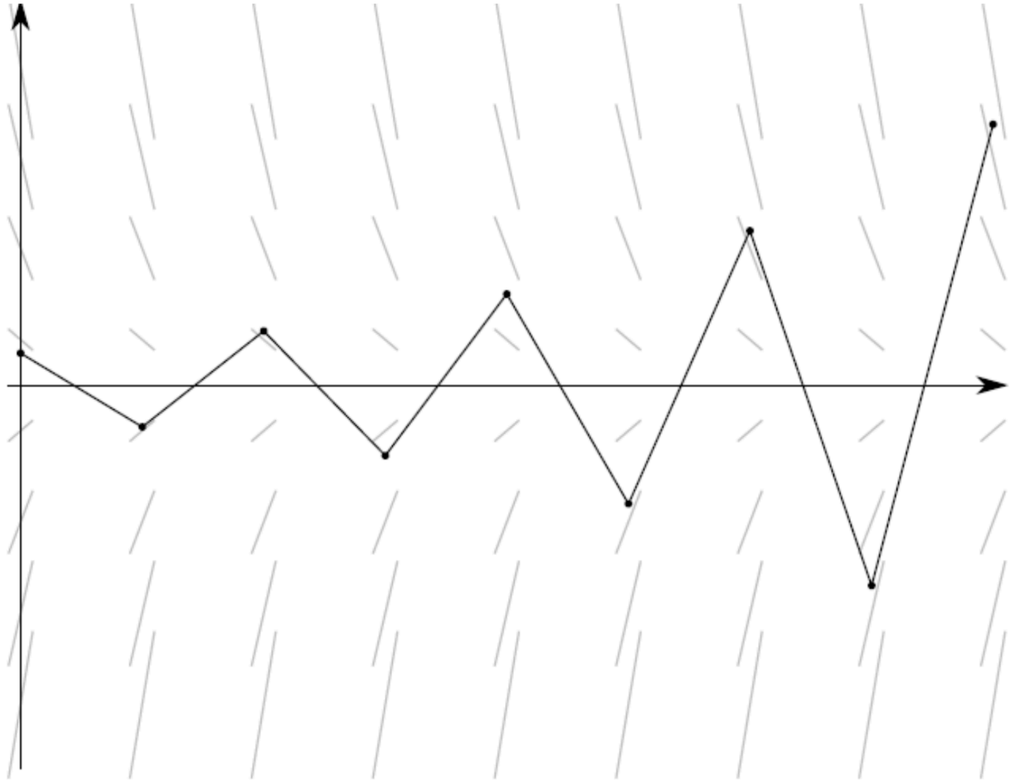
- Explicit method
- $O(h^2)$ localized truncation error
- $O(h)$ global truncation error;
  "first order accurate"

Note: Take $O\left(\frac{1}{h}\right)$ steps to advance $O(1)$ time.

Forward Euler: Stability on model problem, $\dot{y} = a\,y$



Stable ($a = -0.4$)



Unstable ($a = -2.3$)

Forward Euler: Stability on model problem, $\dot{y} = a\,y$

$$y' = ay \longrightarrow y_{k+1} = (1 + ah)y_k$$

For $a < 0$, stable when $h < \frac{2}{|a|}$.

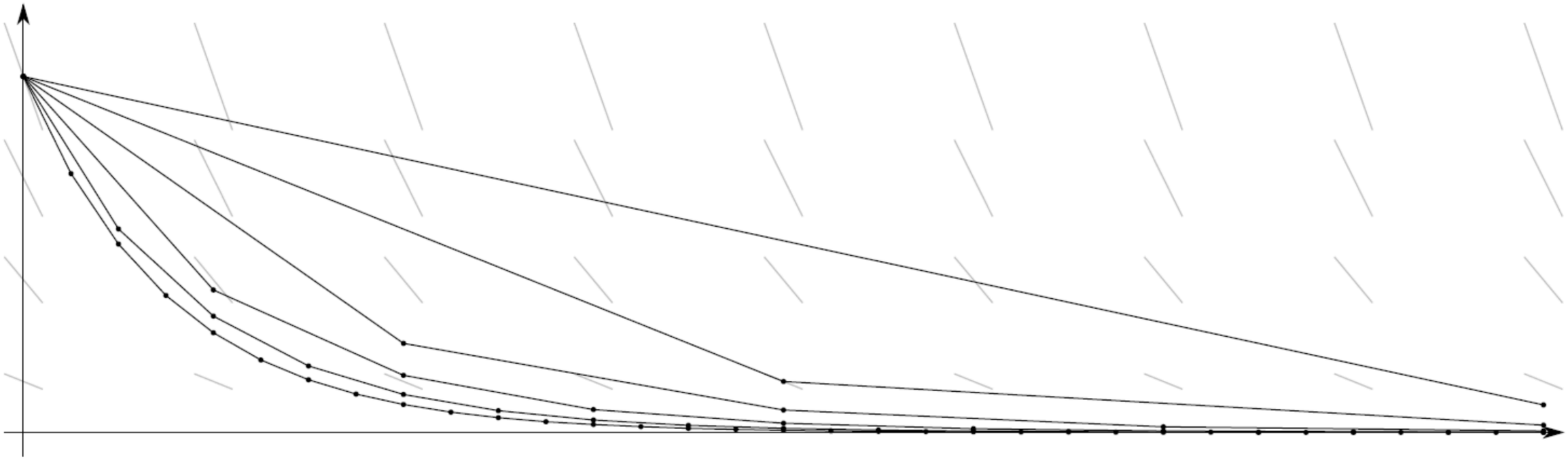
Forward Euler has a well-known time-step restriction for stability.

Backward Euler (a.k.a. Implicit Euler)

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\,\mathbf{F}(\mathbf{y}_{k+1})$$

- Implicit method
- $O(h^2)$ localized truncation error
- $O(h)$ global truncation error;
  "first order accurate"

Backward Euler: Stability on model problem, $\dot{y} = a\,y$



Backward Euler: Stability on model problem, $\dot{y} = a\,y$

$$y' = ay \longrightarrow y_{k+1} = \frac{1}{1-ah} y_k$$

Unconditionally stable!

But this has nothing to do with accuracy.

Good for *stiff* equations.

Numerical Stiffness for IVP ODEs

An IVP is said to be numerically "stiff" if stability requirements dictate a much smaller time step size than is needed to satisfy the approximation requirements alone.

[Ascher & Petzold 1998]

Midpoint Method

$$\mathbf{y}_{k+\frac{1}{2}} = \mathbf{y}_k + \frac{h}{2}\mathbf{F}(\mathbf{y}_k)$$ **Forward Euler half-step**

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\,\mathbf{F}(\mathbf{y}_{k+\frac{1}{2}})$$ **Full-step w/ mid-point gradient**

- Explicit method
- Two function evaluations, but ...
- $O(h^3)$ localized truncation error
- $O(h^2)$ global truncation error
  - "second-order accurate"

Midpoint Method: Stability on model problem, $\dot{y} = a\,y$

$$y_{k+1} = y_k + h\left(a y_{k+\frac{1}{2}}\right) = y_k + ha\left(y_k + \frac{h}{2} a y_k\right) = \left(1 + ha + \frac{h^2 a^2}{2}\right) y_k$$

- Stable when $|y_{k+1}| \leq |y_k|$ or $\left|1 + ha + \frac{h^2 a^2}{2}\right| \leq 1$
- Requires $h \leq \frac{2}{|a|}$ time-step restriction (same as forward Euler).

Symplectic Euler Method (a.k.a. semi-implicit Euler)

- So far time-stepping schemes work with $\mathbf{y} = \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \end{pmatrix}$, and treat $\mathbf{p}$ and $\mathbf{v}$ similarly.
- Downside: Big forces update $\mathbf{v}$ immediately, but $\mathbf{p}$ only sees them on the next timestep.
  - Bad for collisions.
- Idea: Update $\mathbf{v}$ first, then use it to update $\mathbf{p}$
- Aside: Could also update $\mathbf{p}$ first, then use it to update $\mathbf{v}$
  - So-called adjoint version
  - Valid approach, but it delays force integration

Symplectic Euler Method (a.k.a. semi-implicit Euler)

$$\mathbf{v}_{k+1} = \mathbf{v}_k + h\,\mathbf{a}(\mathbf{p}_k, \mathbf{v}_k) \qquad \textbf{Update velocity}$$

$$\mathbf{p}_{k+1} = \mathbf{p}_k + h\,\mathbf{v}_{k+1} \qquad \textbf{Update position}$$

- Semi-implicit method
- One function evaluation
- $O(h^2)$ localized truncation error
- $O(h)$ global truncation error
  - "first-order accurate"
- Symplectic structure
  - preserves area in position-momentum phase space

Symplectic Euler Method: Stability on model problem, $\dot{y} = a\,y$

N/A

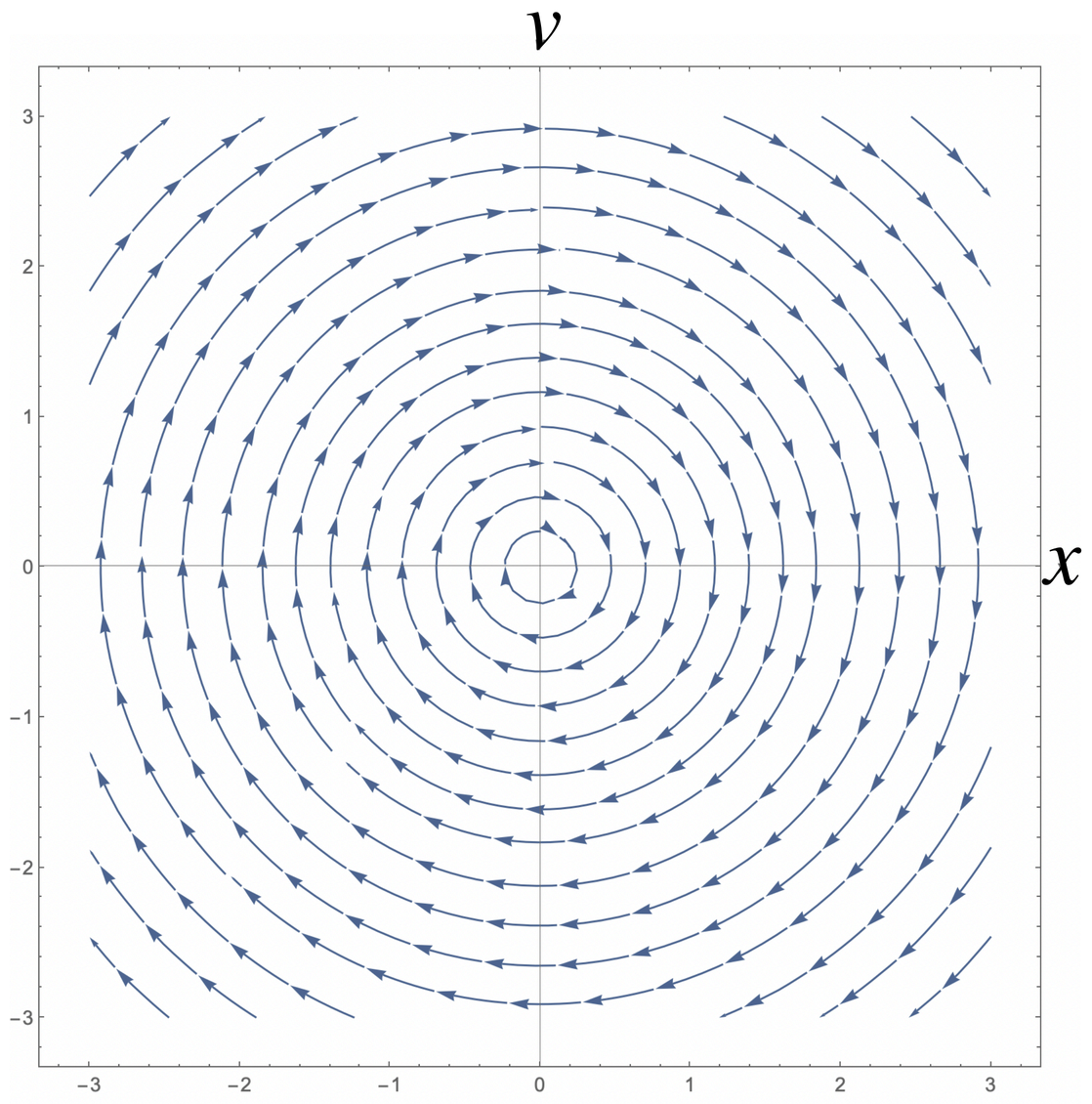
Harmonic oscillator model problem

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v \\ -x \end{pmatrix} \quad \Longleftrightarrow \quad \dot{\mathbf{r}} = \mathbf{r}_\perp$$

- **Harmonic oscillator equation**

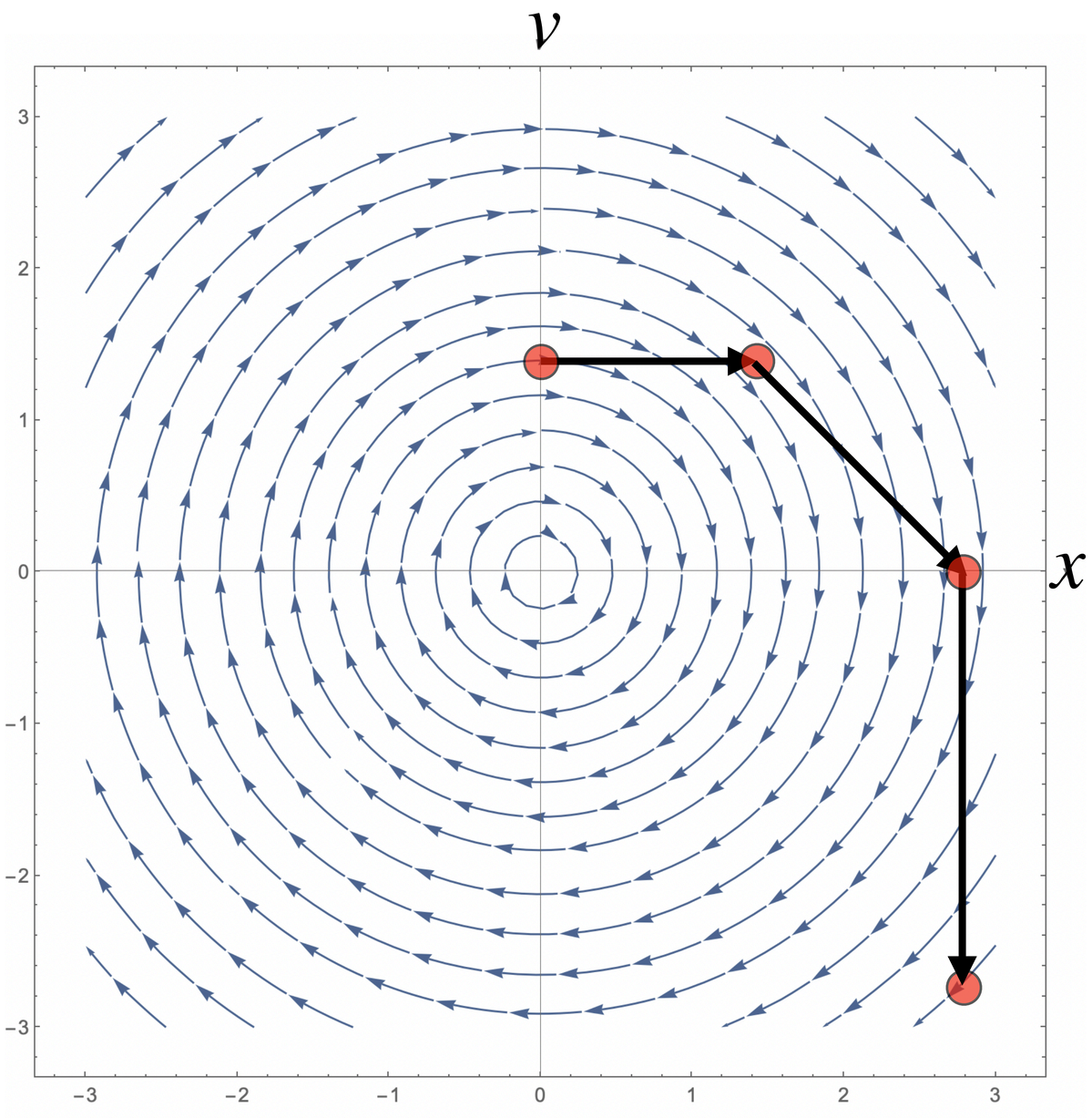
$$\ddot{x} + \omega^2 x = 0$$

with unit natural frequency, $\omega = 1.$



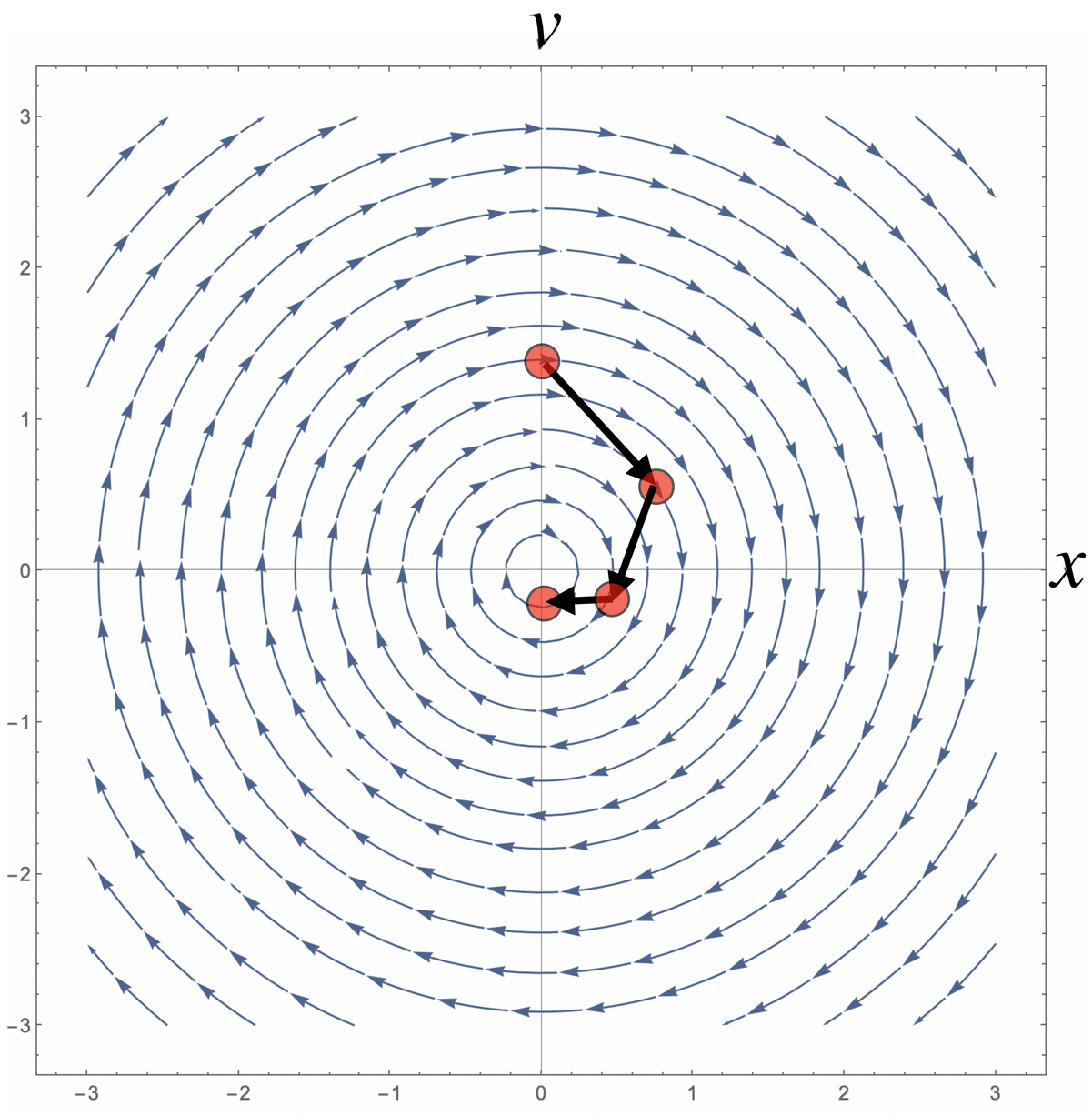
Stability on harmonic oscillator model problem

Forward Euler

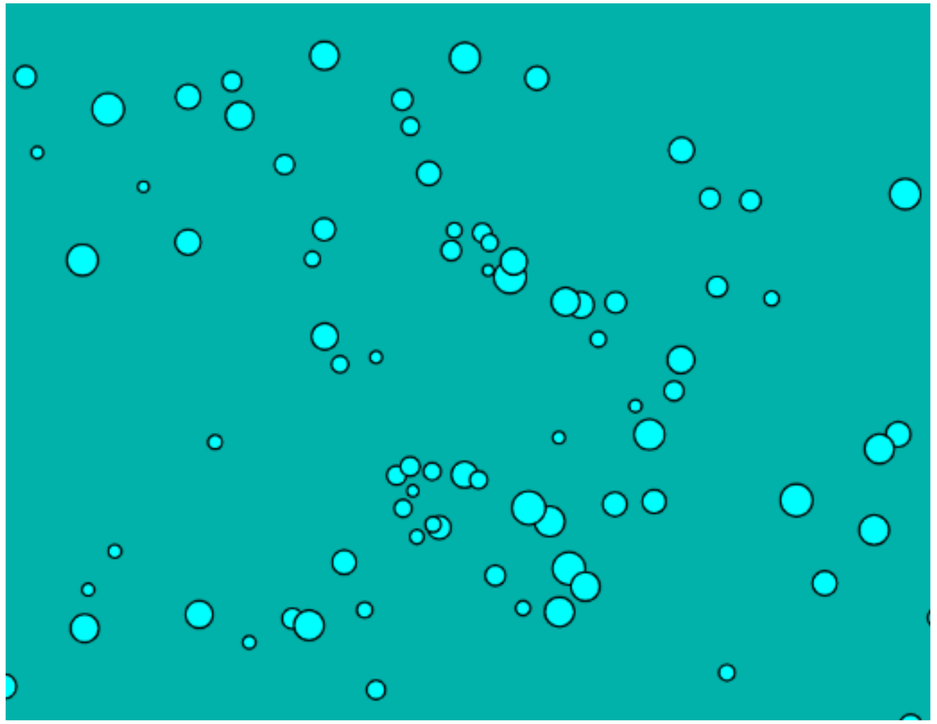


Stability on harmonic oscillator model problem

Backward Euler



Example: Particle advection on vortex



https://www.openprocessing.org/sketch/966498

Stability on harmonic oscillator model problem

Forward vs Symplectic Euler

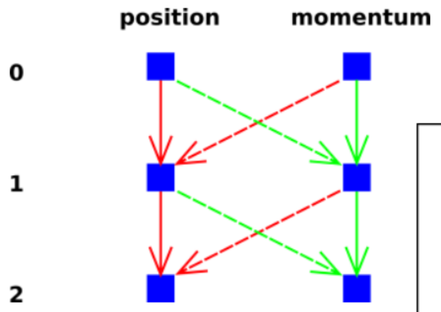


Figure 5: Forward Euler – Criss-Cross

Forward Euler

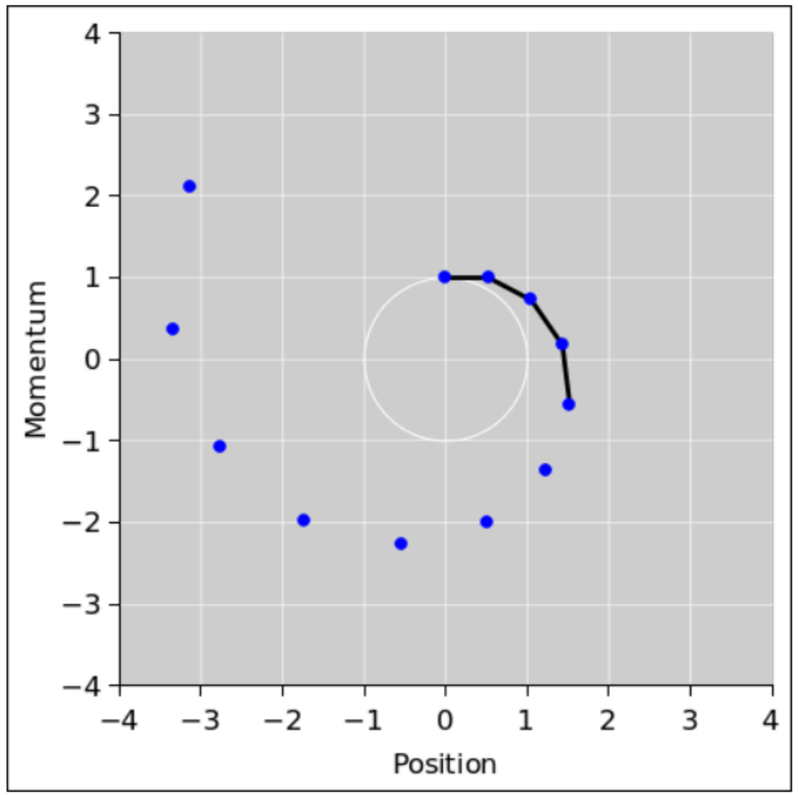


Figure 7: Non-Symplectic First-Order Euler Steps

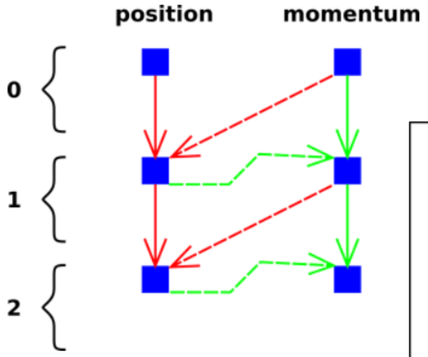


Figure 4: Symplectic First-Order Euler

Symplectic Euler

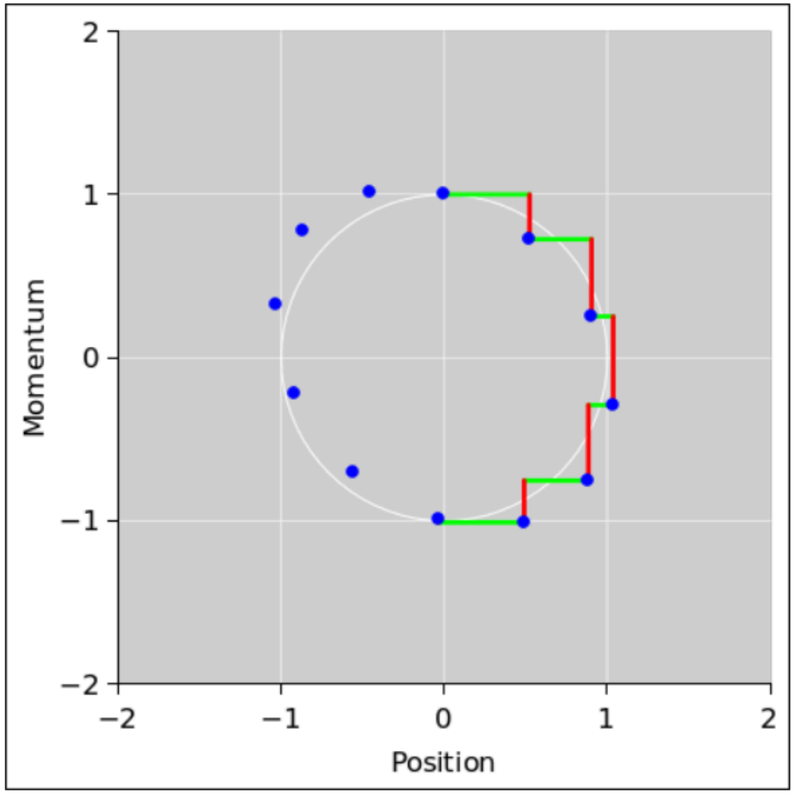


Figure 6: Symplectic First-Order Euler Steps

https://www.av8n.com/physics/symplectic-integrator.htm

Symplectic Integrators

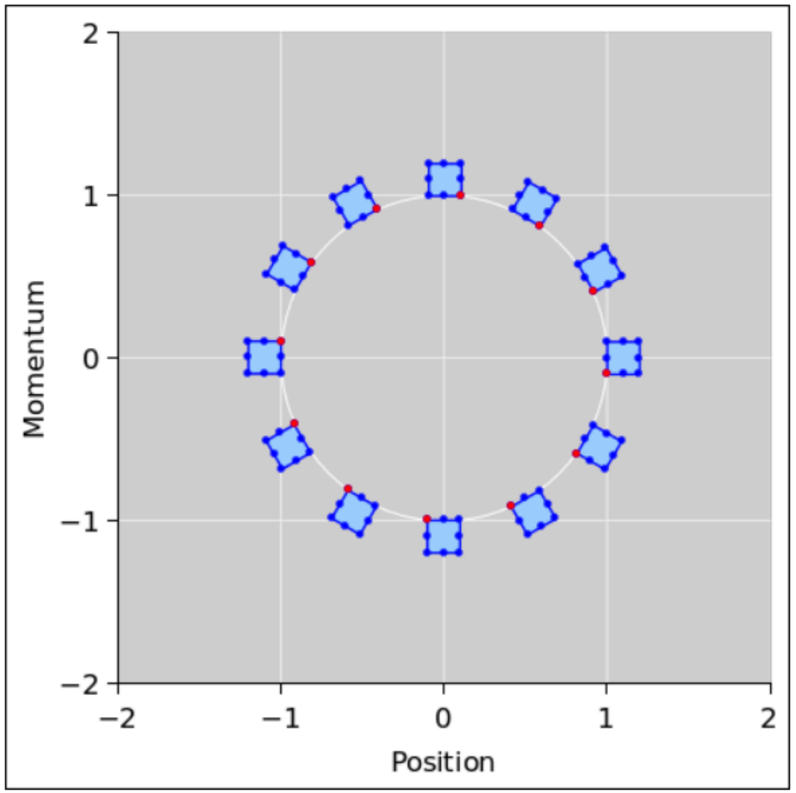


Figure 1: Phase Space of a Harmonic Oscillator

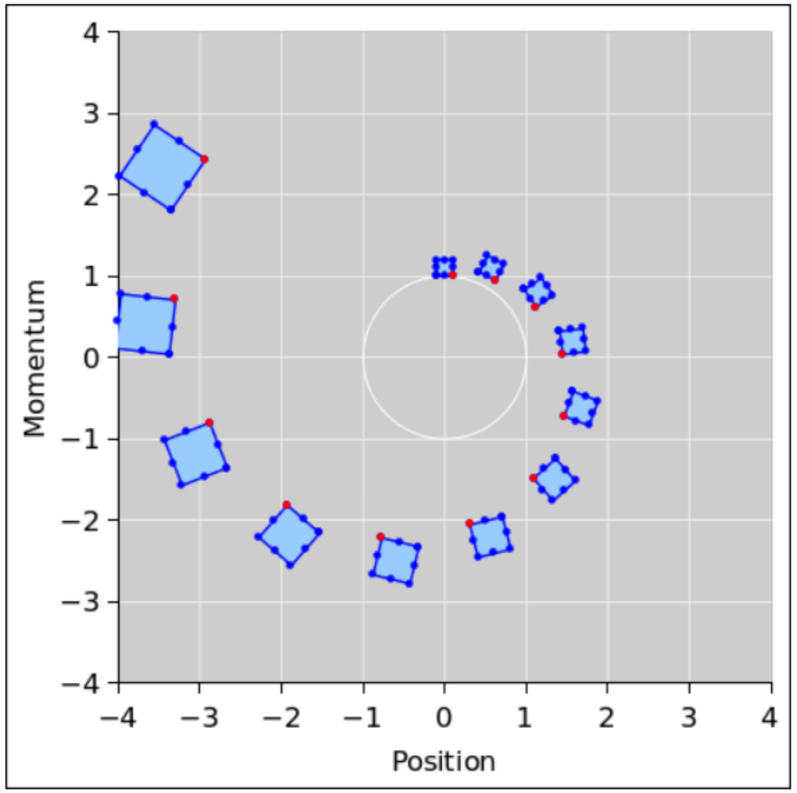


Figure 3: Non-Symplectic Integration : Harmonic Oscillator

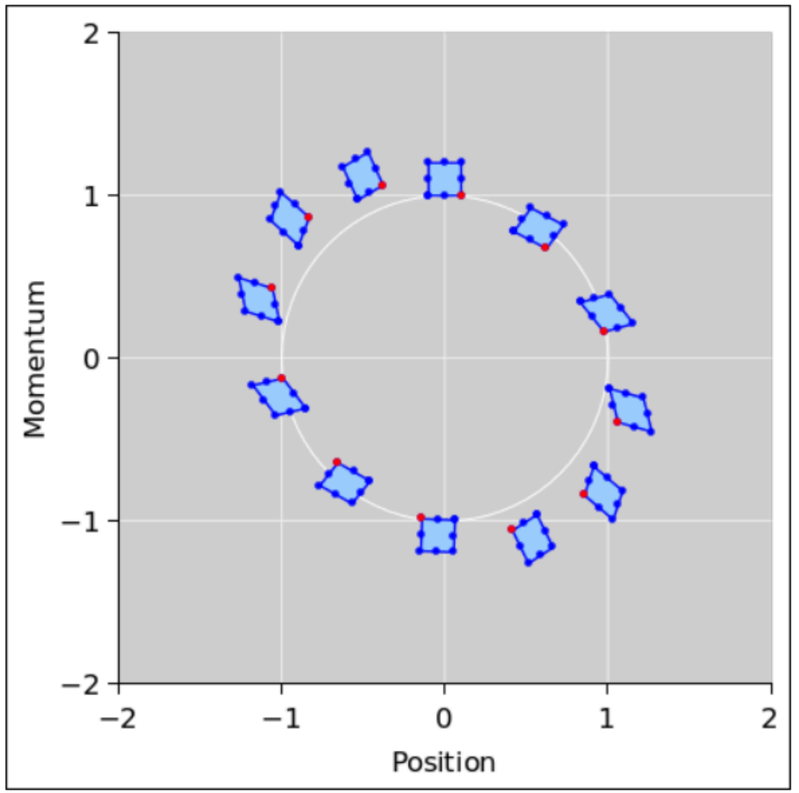
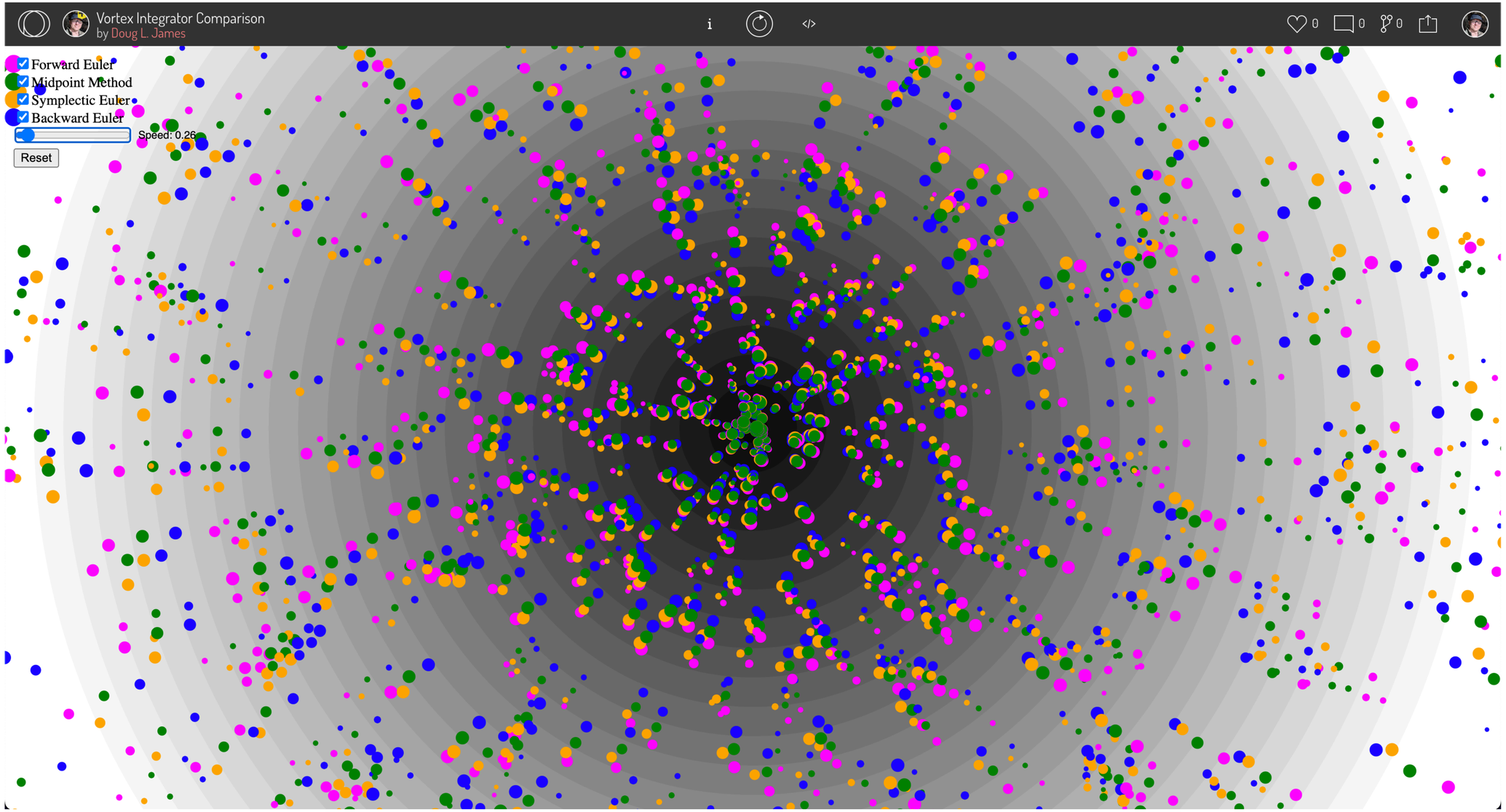


Figure 2: Symplectic Integration : Harmonic Oscillator

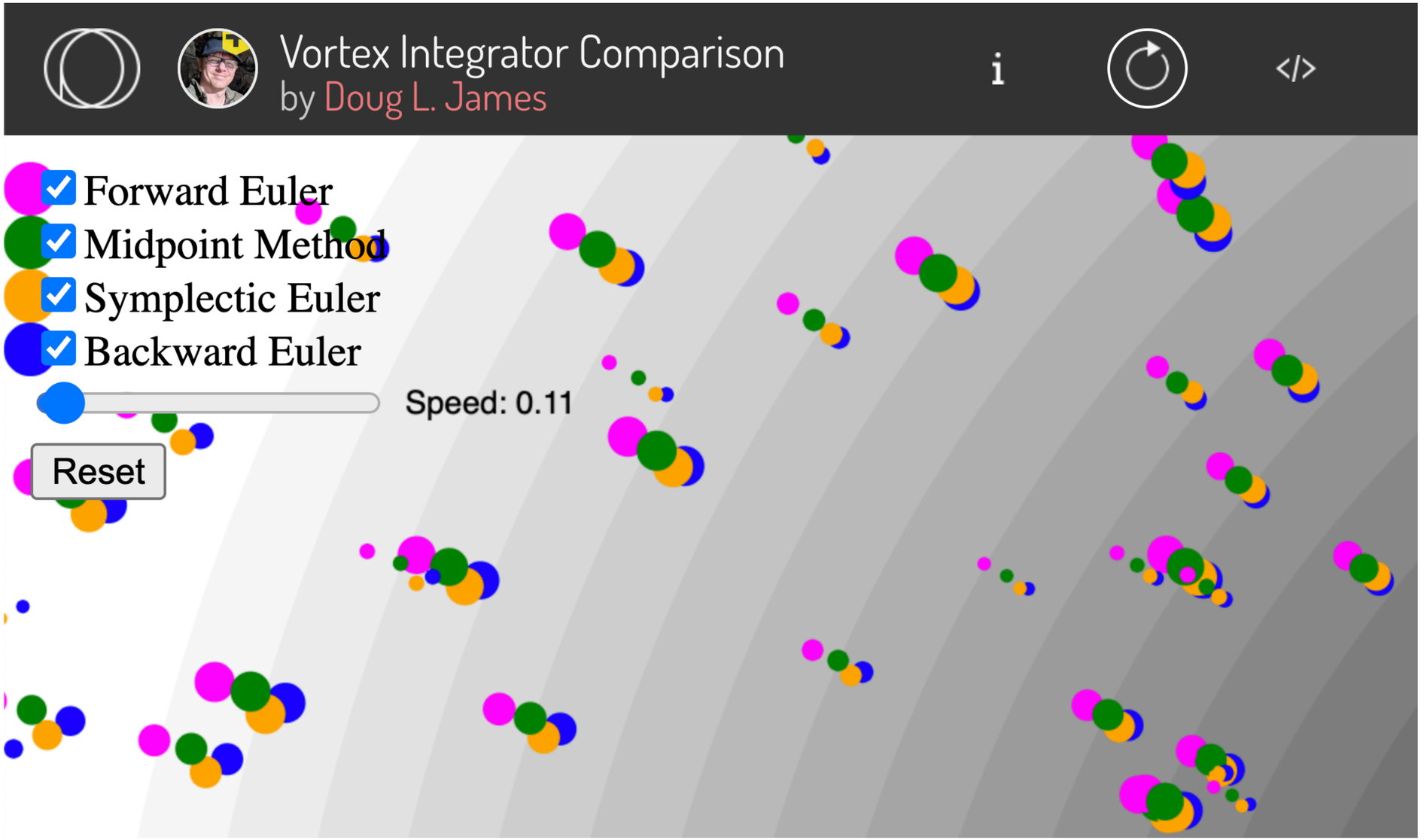
https://www.av8n.com/physics/symplectic-integrator.htm

Integrator Comparison for Circular Motion



https://openprocessing.org/sketch/2029371

Integrator Comparison for Circular Motion



https://openprocessing.org/sketch/2029371