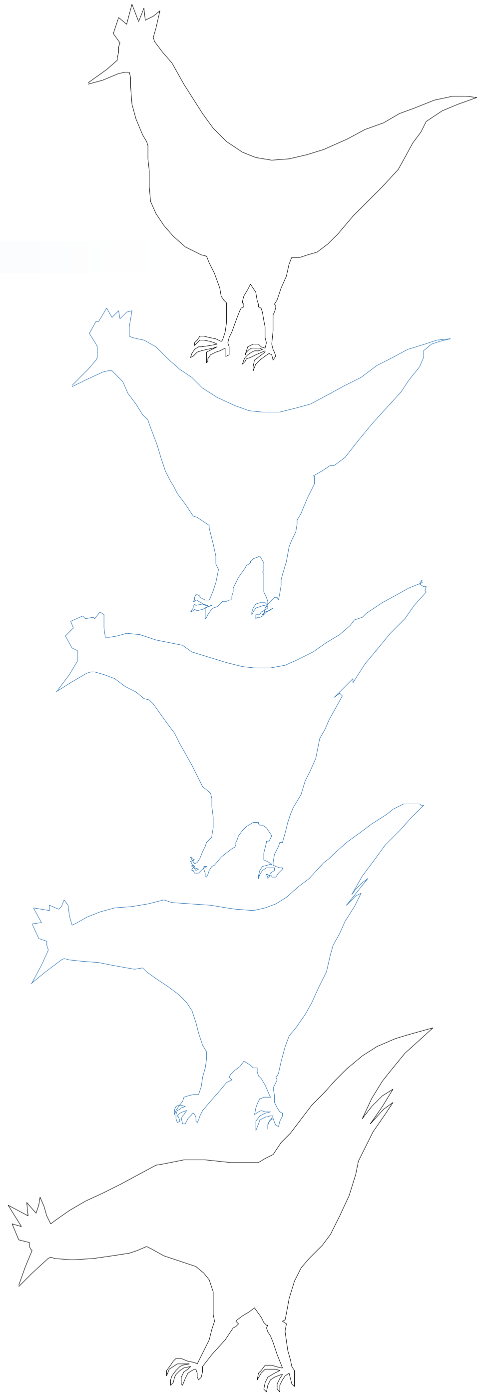
# Particle Systems

CS418 Interactive Computer Graphics

John C. Hart

# Flexible Body Animation

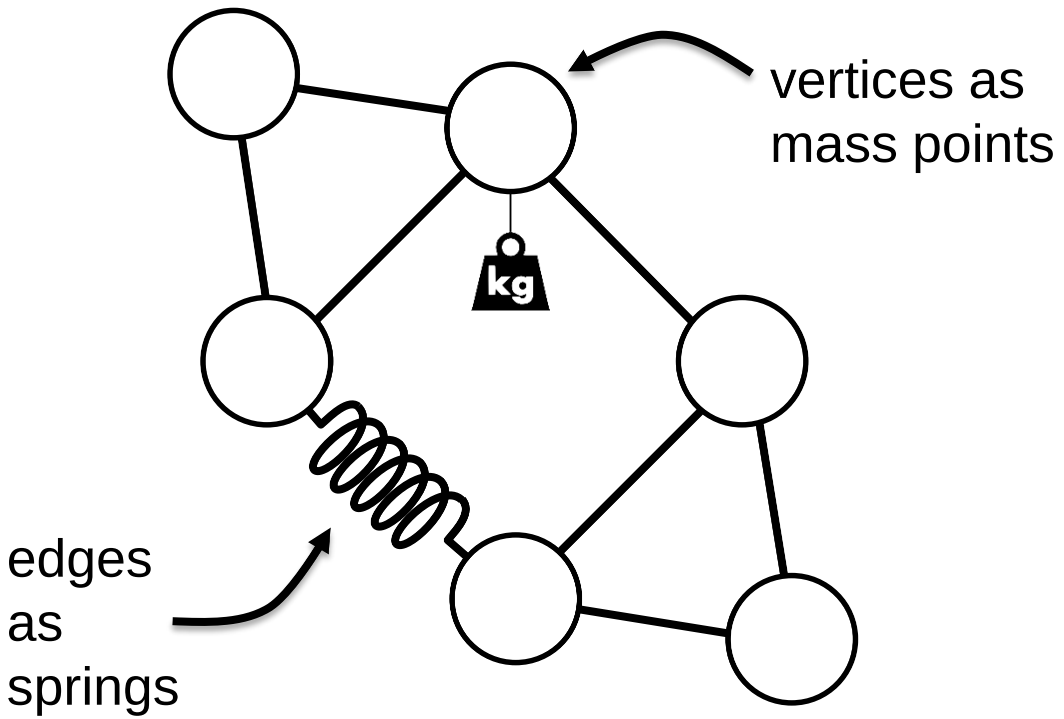
- Need same number and configuration of vertices at key frames for intervening frames to make sense
- Need to have correspondences between two collections of vertices
- Vertices → Particles
- Edges → Springs
- Moving a vertex drags and pushes other vertices into position from tension and compression on the springs



A motivating example from: Sederberg & Greenwood, A Physically-Based Approach to 2-D Shape Blending, Proc. SIGGRAPH 92

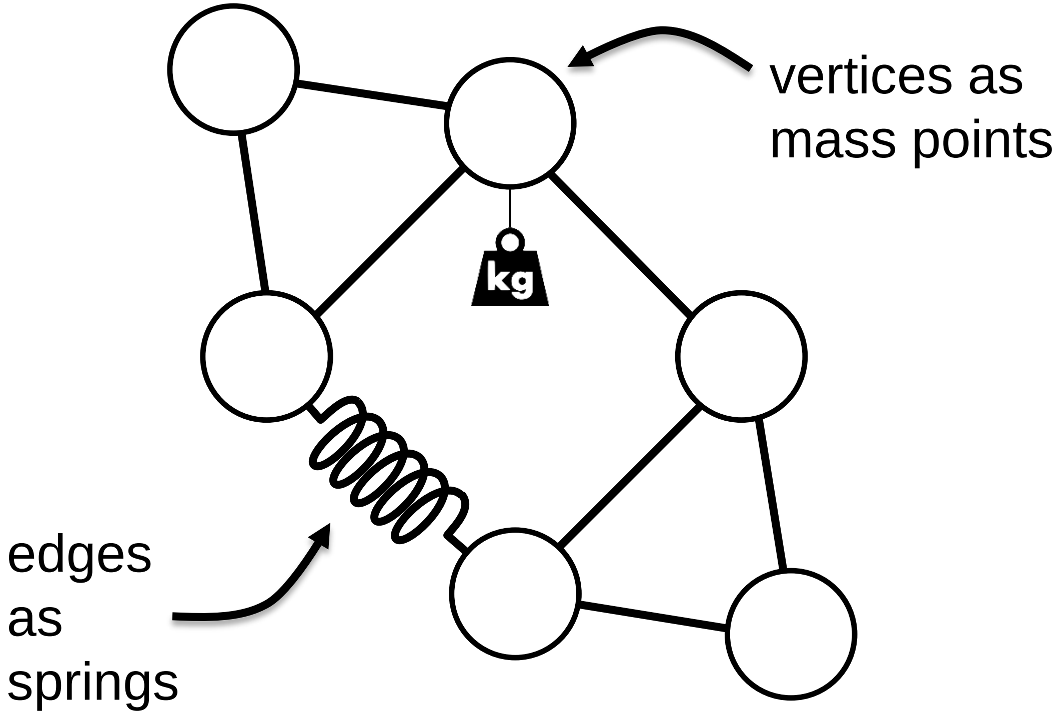
# Physically-Based Modeling

- Animate shapes by kinematic simulation



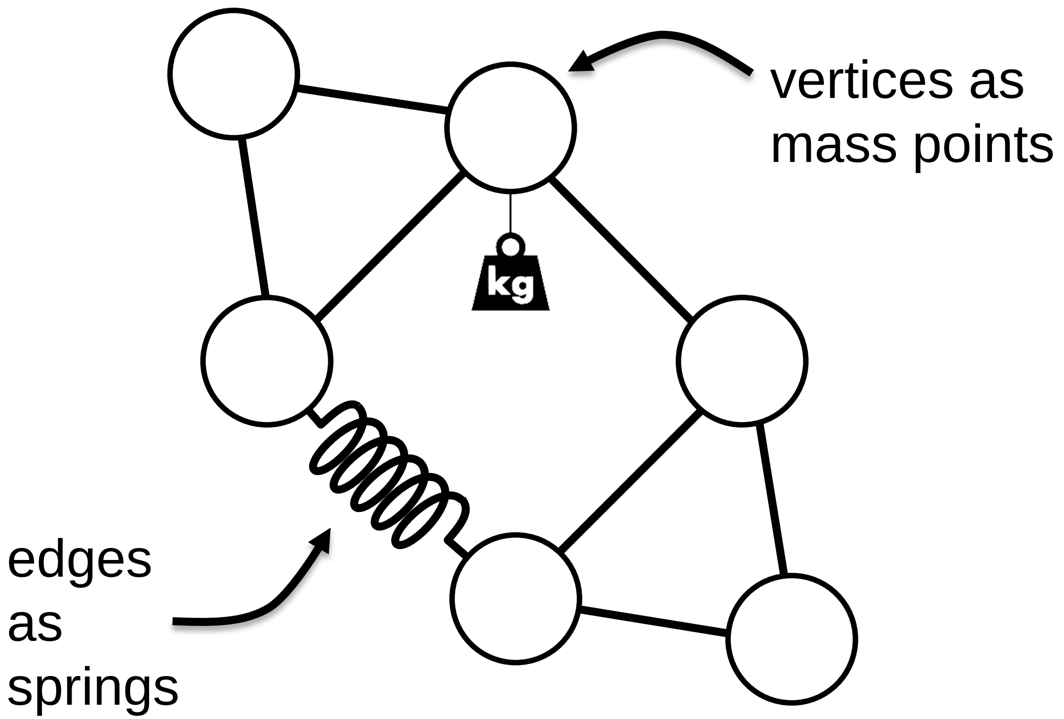
# Physically-Based Modeling

- Animate shapes by kinematic simulation
  - Newton: $\mathbf{F} = m\mathbf{a}$



# Physically-Based Modeling

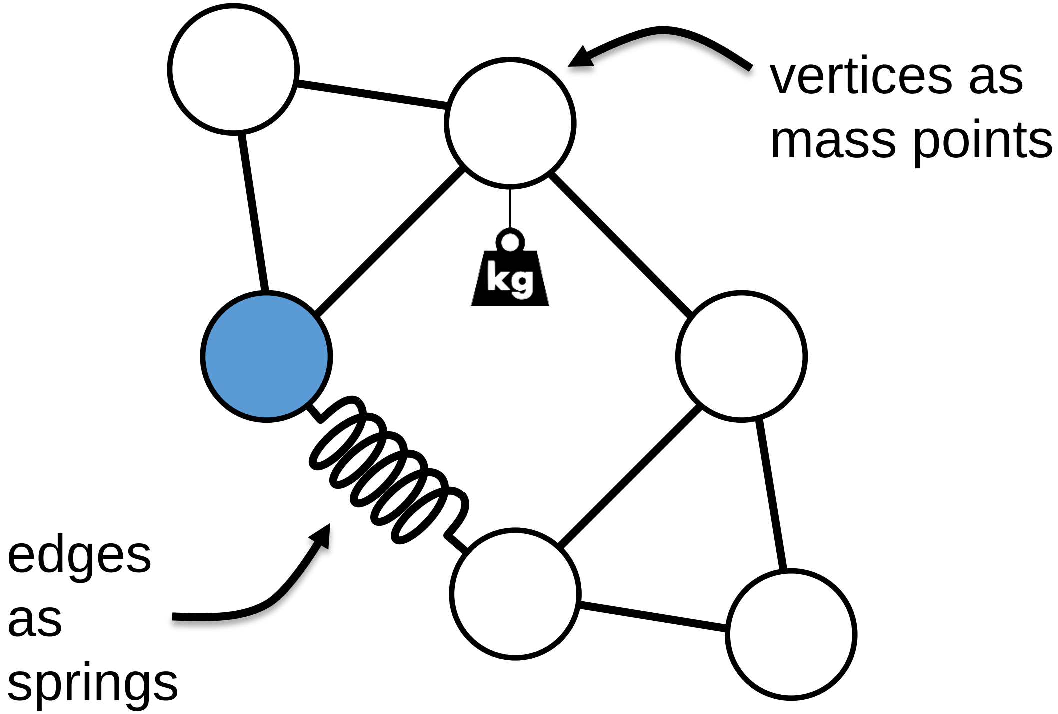
- Animate shapes by kinematic simulation
  - Newton: $\mathbf{F} = m\mathbf{a}$
  - Aristotle: $\mathbf{F} = m\mathbf{v}$



# Physically-Based Modeling

- Animate shapes by kinematic simulation
  - Newton: $\mathbf{F} = m\mathbf{a}$
  - Aristotle: $\mathbf{F} = m\mathbf{v}$

- Particles
  - Position: $\mathbf{x} = (x, y, z, \ldots)$
  - …over time: $\mathbf{x}(t) = (x(t), y(t), z(t), \ldots)$
  - Velocity: $\mathbf{v} = \mathbf{x}' = d\mathbf{x}dt = f(\mathbf{x}, t)$

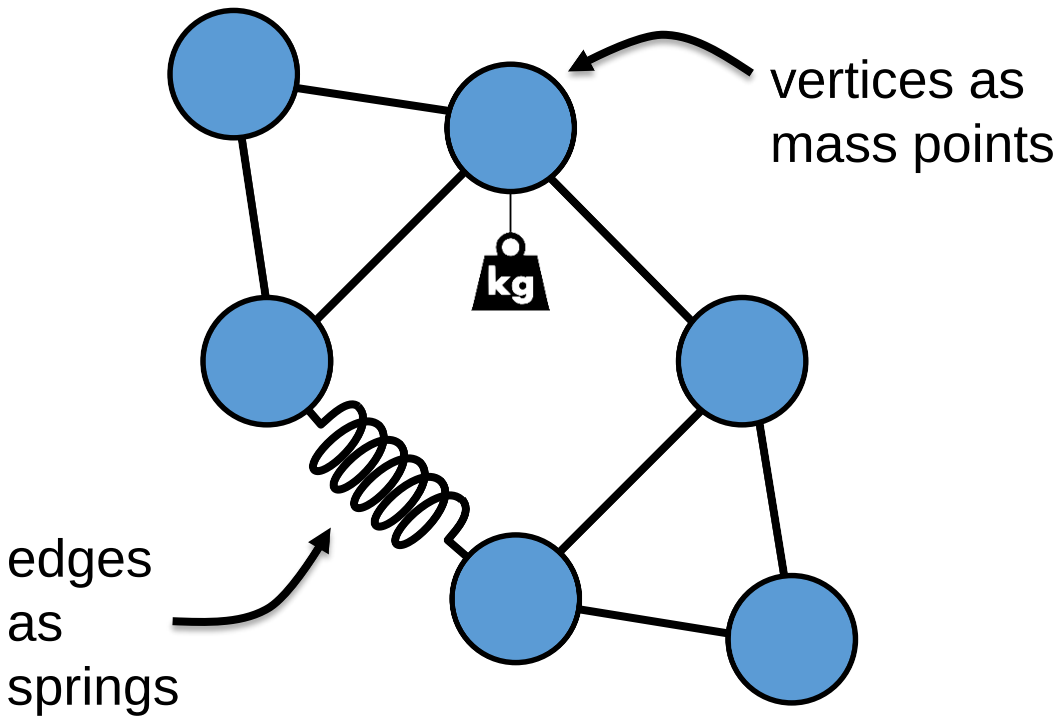


# Physically-Based Modeling

- Animate shapes by kinematic simulation
  - Newton: $\mathbf{F} = m\mathbf{a}$
  - Aristotle: $\mathbf{F} = m\mathbf{v}$

- Particles
  - Position: $\mathbf{x} = (x, y, z, \ldots)$
  - …over time: $\mathbf{x}(t) = (x(t), y(t), z(t), \ldots)$
  - Velocity: $\mathbf{v} = \mathbf{x}' = d\mathbf{x}dt = f\,(\mathbf{x}, t)$

- Shape described by $\mathbf{x}$

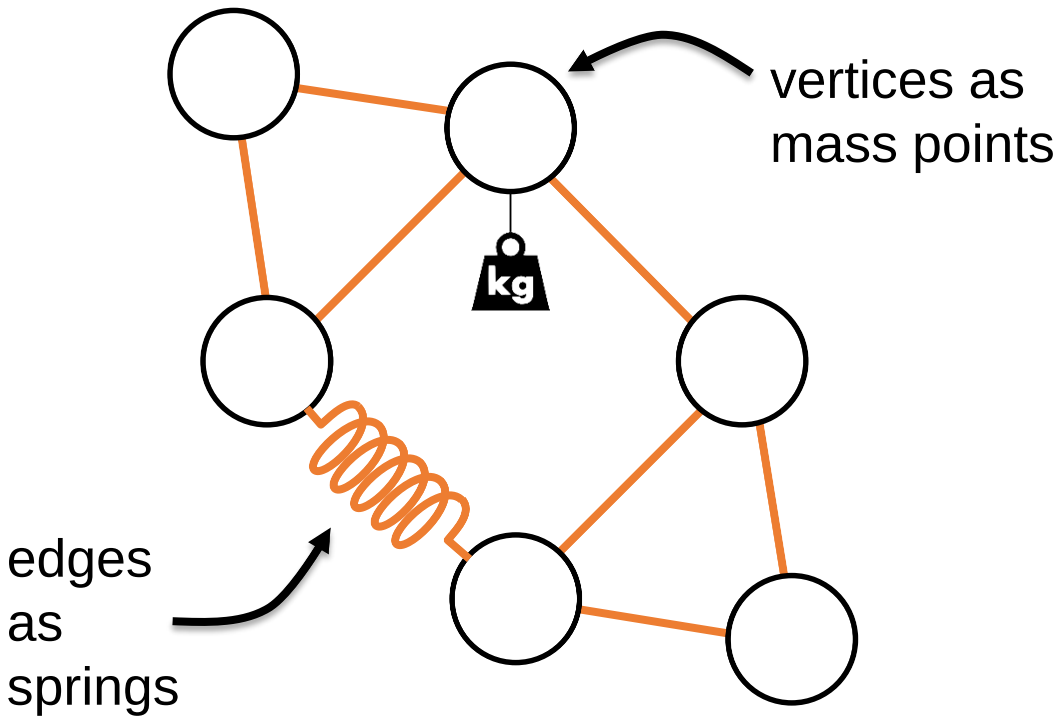


# Physically-Based Modeling

- Animate shapes by kinematic simulation
  - Newton: $\mathbf{F} = m\mathbf{a}$
  - Aristotle: $\mathbf{F} = m\mathbf{v}$

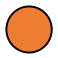
- Particles
  - Position: $\mathbf{x} = (x, y, z, \ldots)$
  - …over time: $\mathbf{x}(t) = (x(t), y(t), z(t), \ldots)$
  - Velocity: $\mathbf{v} = \mathbf{x}' = d\mathbf{x}dt = f(\mathbf{x}, t)$

- Shape described by $\mathbf{x}$
- Behavior described by $f()$



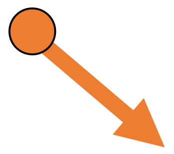
# Differential Equations

- Moving point: $\mathbf{x}(t)$



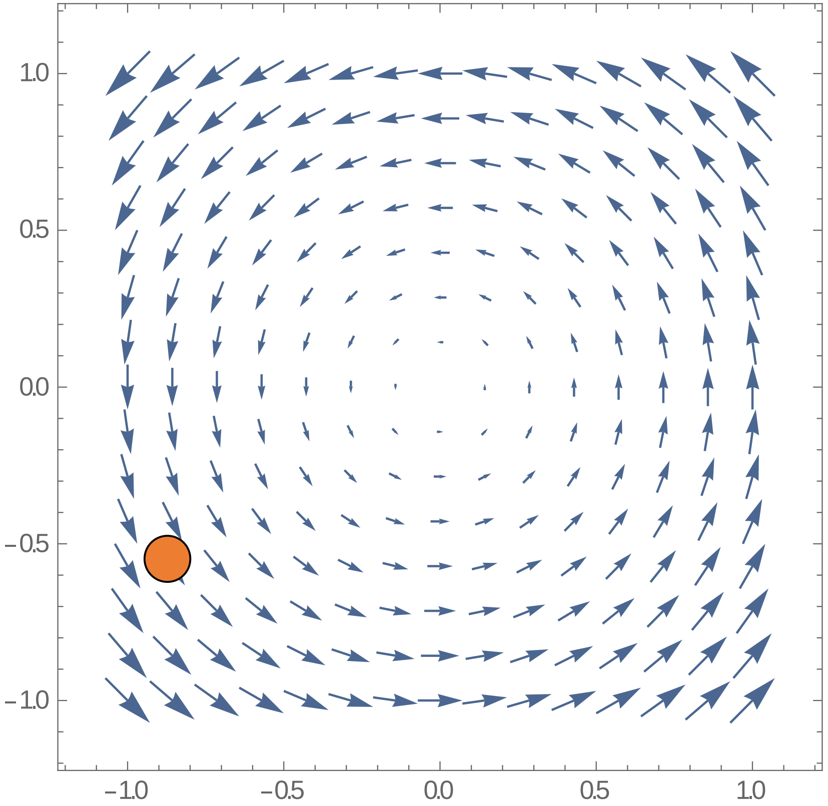
# Differential Equations

- Moving point: $\mathbf{x}(t)$
- Velocity: $\mathbf{x}'$



# Differential Equations

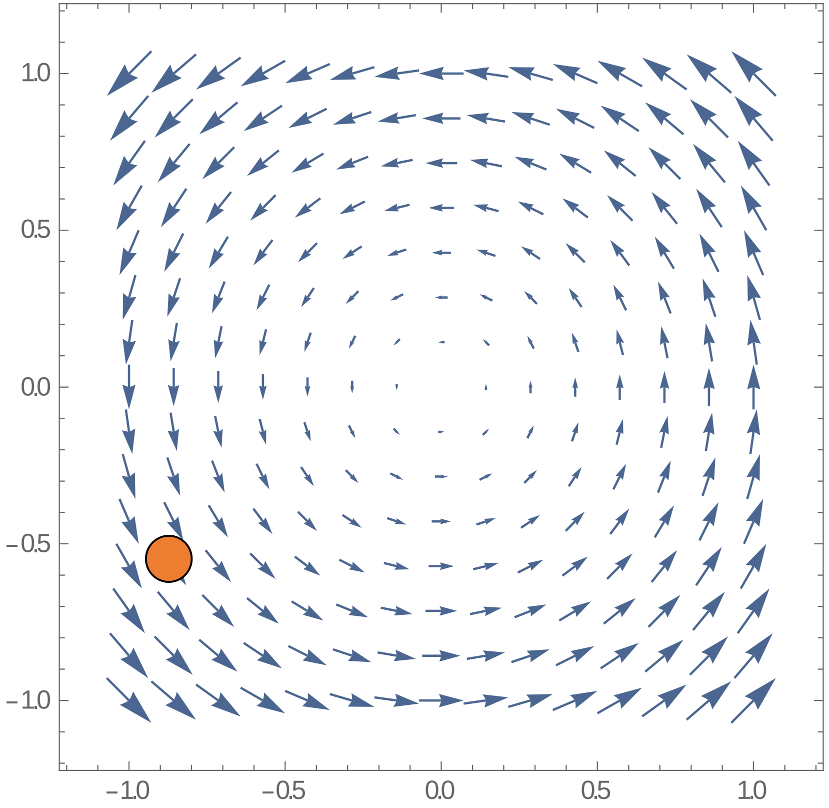
- Moving point: $\mathbf{x}(t)$
- Velocity: $\mathbf{x}' = f(\mathbf{x}, t)$




$f((x,y),t) = (-y,x)$

# Differential Equations

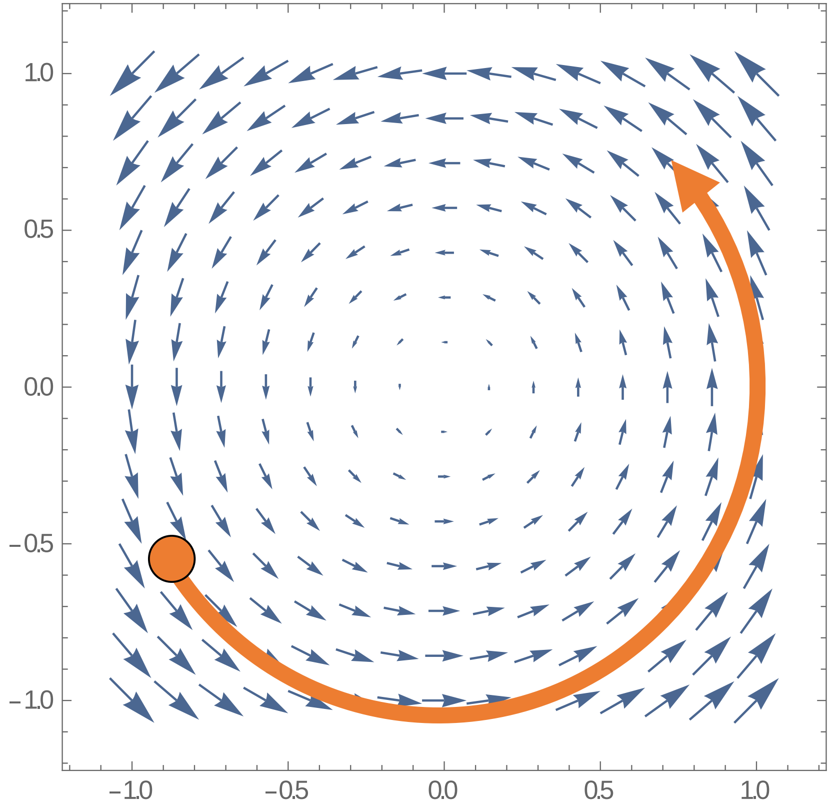
- Moving point: $\mathbf{x}(t)$
- Velocity: $\mathbf{x}' = f(\mathbf{x},t)$
- Initial value problem
  - Given position $\mathbf{x}(0)$
  - Where is $\mathbf{x}(t)$?




$f((x,y),t) = (-y,x)$

# Differential Equations

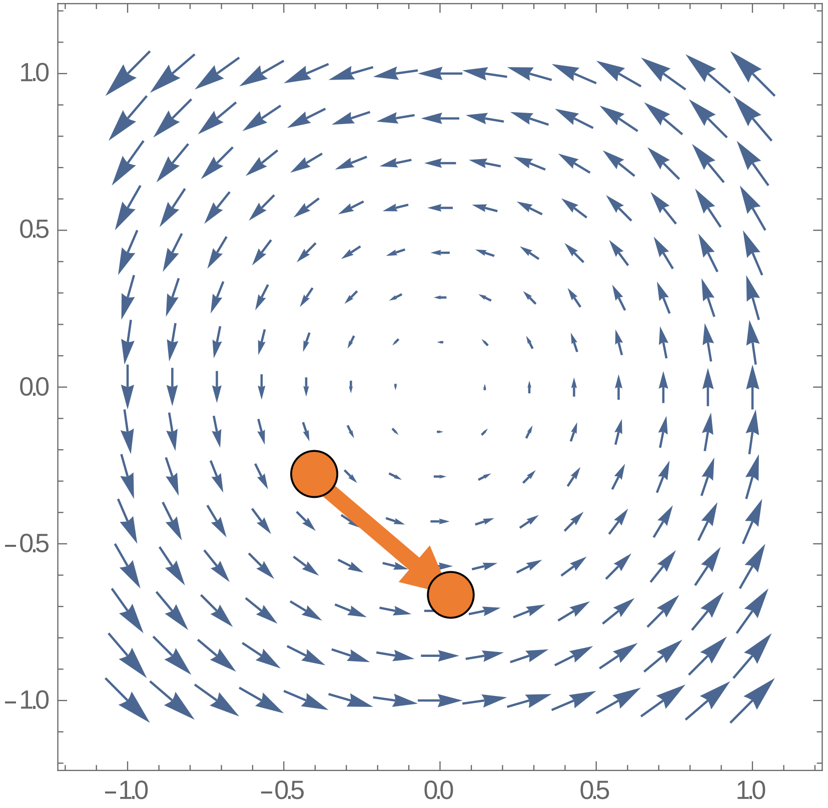
- Moving point: **x**($t$)
- Velocity: **x'** = $f$ (**x**,$t$)
- Initial value problem
  - Given position **x**(0)
  - Where is **x**($t$)?




$f$ ((x,y),$t$) = (-y,x)

# Euler Integration

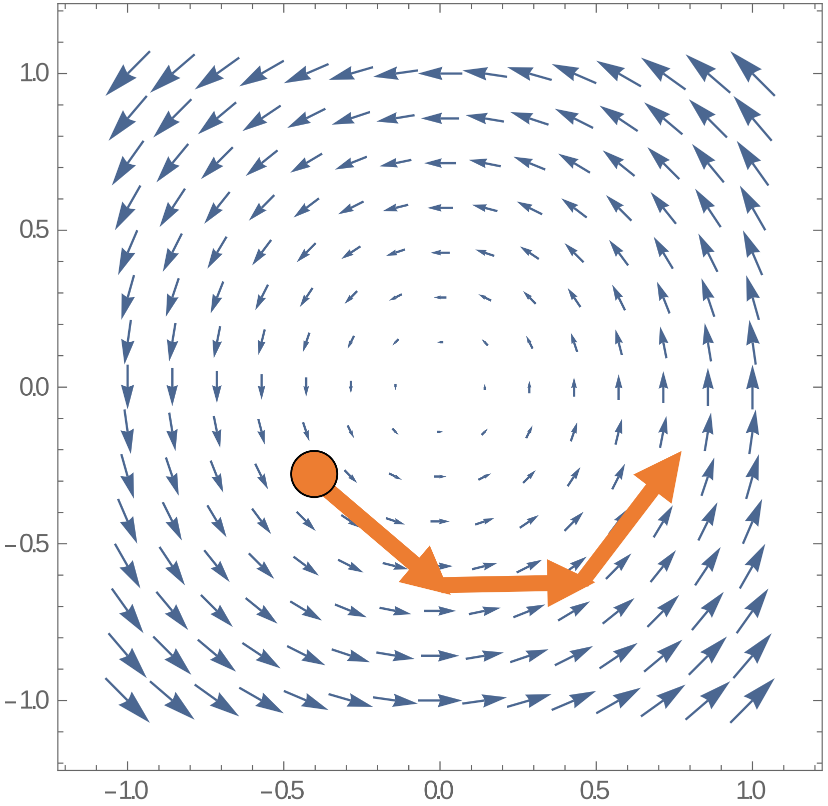
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, f\,(\mathbf{x}(t), t)$$




$f\,((x,y),t) = (-y,x)$

# Euler Integration

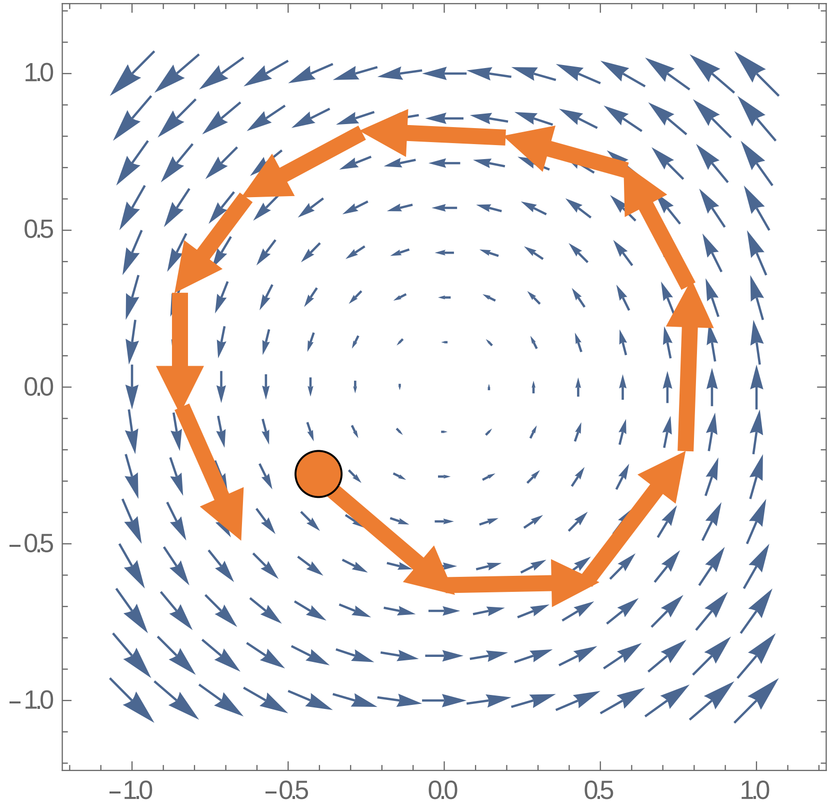
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, f(\mathbf{x}(t), t)$$




$f((x,y),t) = (-y,x)$

# Euler Integration

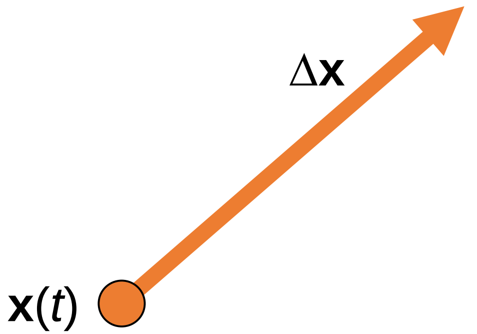
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, f\,(\mathbf{x}(t), t)$$




$f\,((x,y),t) = (-y,x)$

# Midpoint Method

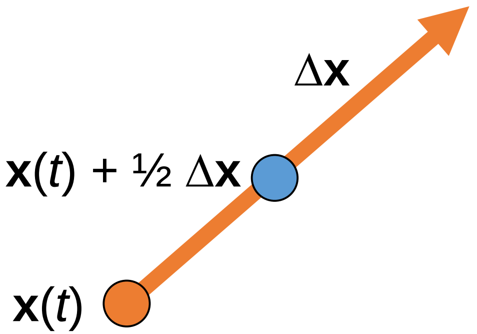
$\Delta \mathbf{x} = \Delta t \, f\,(\mathbf{x}(t), t)$




$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t \, f\,(\mathbf{x}(t) + \tfrac{1}{2}\, \Delta \mathbf{x}, t \;+ \tfrac{1}{2}\, \Delta t)$

# Midpoint Method

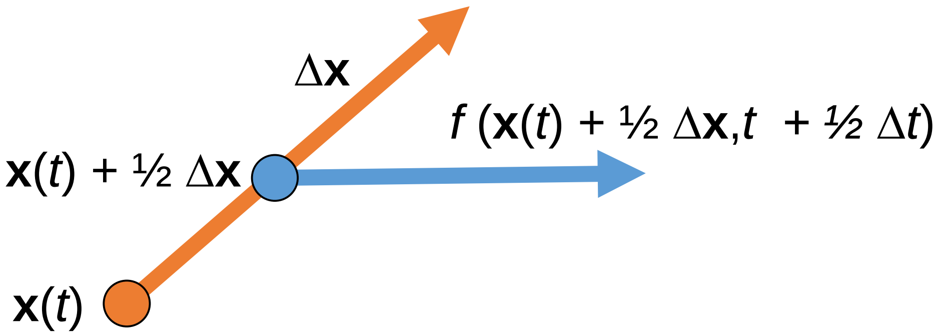
$$\Delta \mathbf{x} = \Delta t\, f(\mathbf{x}(t), t)$$




$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, f(\mathbf{x}(t) + \tfrac{1}{2}\Delta \mathbf{x}, t + \tfrac{1}{2}\Delta t)$$

# Midpoint Method

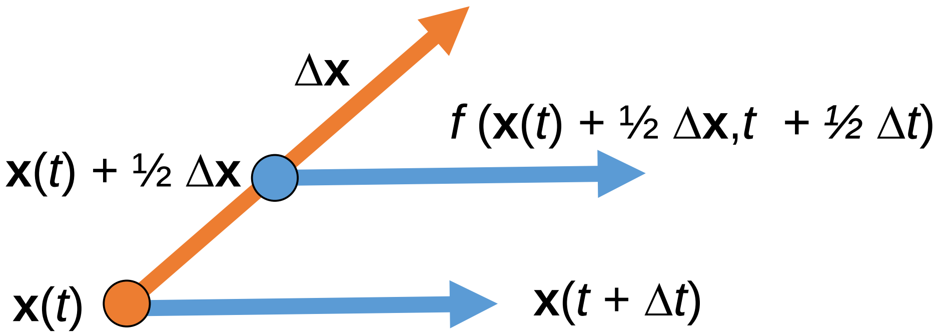
$$\Delta \mathbf{x} = \Delta t\, f\,(\mathbf{x}(t), t)$$




$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, f\,(\mathbf{x}(t) + \tfrac{1}{2} \Delta \mathbf{x}, t + \tfrac{1}{2} \Delta t)$$

# Midpoint Method

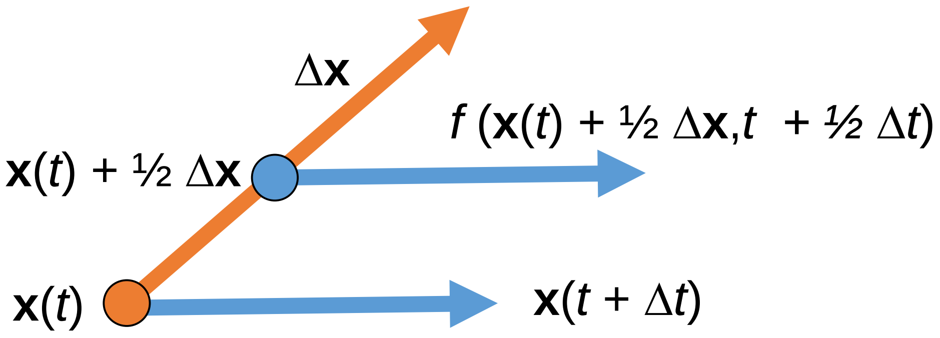
$$\Delta \mathbf{x} = \Delta t\, f\,(\mathbf{x}(t), t)$$




$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, f\,(\mathbf{x}(t) + \tfrac{1}{2}\Delta \mathbf{x}, t + \tfrac{1}{2}\Delta t)$$

# Midpoint Method

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

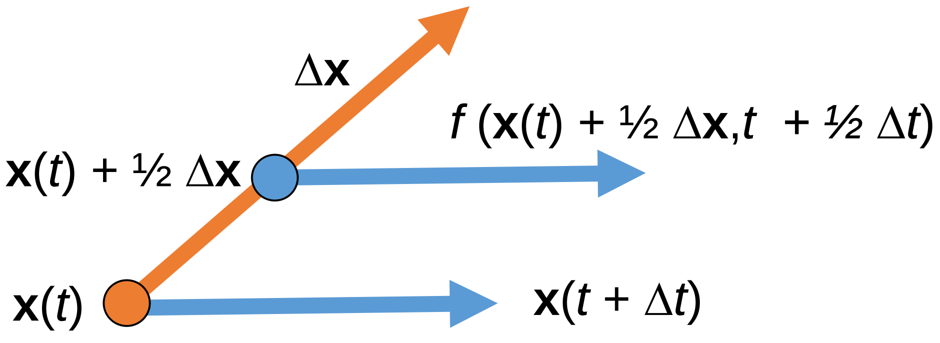



$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t \, f\,(\mathbf{x}(t) + \tfrac{1}{2}\,\Delta \mathbf{x}, t + \tfrac{1}{2}\,\Delta t)$

- Also higher order Runge-Kutta methods

# Midpoint Method

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

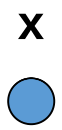



$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, f(\mathbf{x}(t) + \tfrac{1}{2}\Delta \mathbf{x}, t + \tfrac{1}{2}\Delta t)$

- Also higher order Runge-Kutta methods
- Need to be able to evaluate $f(\mathbf{x}, t)$ anywhere and anytime

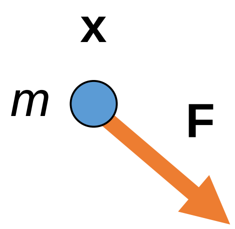
# One Lousy Particle

- Position: $\mathbf{x} = (x, y, z, \ldots)$



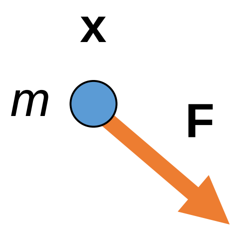
# One Lousy Particle

- Position: $\mathbf{x} = (x, y, z, \ldots)$
- Velocity: $\mathbf{v} = \mathbf{x}' = d\mathbf{x}dt$
- Acceleration: $\mathbf{a} = \mathbf{x}'' = d\mathbf{v}dt$
- Newton: $\mathbf{F} = m\mathbf{a}$



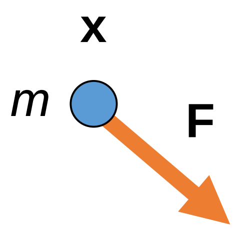
# One Lousy Particle

- Position: $\mathbf{x} = (x, y, z, \ldots)$
- Velocity: $\mathbf{v} = \mathbf{x}' = d\mathbf{x}dt$
- Acceleration: $\mathbf{a} = \mathbf{x}'' = d\mathbf{v}dt$
- Newton: $\mathbf{F} = m\mathbf{a}$
- Need to integrate $\mathbf{x}'' = \mathbf{F}/m = f\,(\mathbf{x},\mathbf{x}',t)$
- Second order differential equation



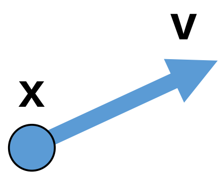
# One Lousy Particle

- Position: $\mathbf{x} = (x, y, z, \ldots)$
- Velocity: $\mathbf{v} = \mathbf{x}' = d\mathbf{x}dt$
- Acceleration: $\mathbf{a} = \mathbf{x}'' = d\mathbf{v}dt$
- Newton: $\mathbf{F} = m\mathbf{a}$
- Need to integrate $\mathbf{x}'' = \mathbf{F}/m = f\,(\mathbf{x},\mathbf{x}',t)$
- Second order differential equation
- We don’t know how to solve a second order differential equation



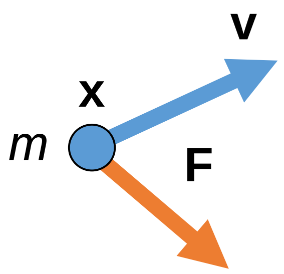
# Phase Space

- Position, velocity: $\mathbf{x} = (\mathbf{p},\mathbf{v}) = (px, py, pz, vx, vy, vz)$



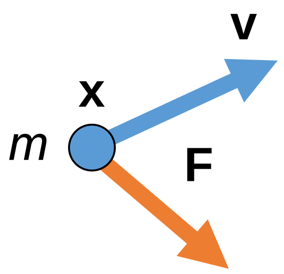
# Phase Space

- Position, velocity: **x** = (**p**,**v**) = (*px*, *py*, *pz*, *vx*, *vy*, *vz*)
- Derivative: **x**’ = *d***x***dt* = (**p**’,**v**’) = (**v**,**a**)



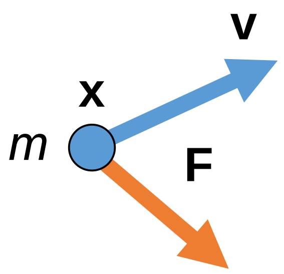
# Phase Space

- Position, velocity: $\mathbf{x} = (\mathbf{p},\mathbf{v}) = (px, py, pz, vx, vy, vz)$
- Derivative: $\mathbf{x}' = d\mathbf{x}dt = (\mathbf{p}',\mathbf{v}') = (\mathbf{v},\mathbf{a})$
- Need to integrate $\mathbf{x}' = f(\mathbf{x}, t)$



# Phase Space

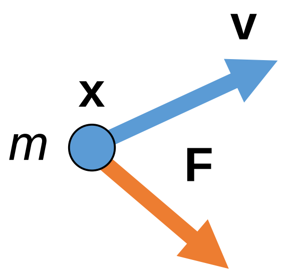
- Position, velocity: $\mathbf{x} = (\mathbf{p},\mathbf{v}) = (px, py, pz, vx, vy, vz)$
- Derivative: $\mathbf{x}' = d\mathbf{x}dt = (\mathbf{p}',\mathbf{v}') = (\mathbf{v},\mathbf{a})$
- Need to integrate $\mathbf{x}' = f(\mathbf{x}, t)$




$$\mathbf{x'} = f((\mathbf{p},\mathbf{v}), \mathrm{t})$$

# Phase Space

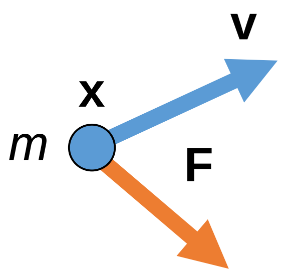
- Position, velocity: $\mathbf{x} = (\mathbf{p},\mathbf{v}) = (px, py, pz, vx, vy, vz)$
- Derivative: $\mathbf{x}' = d\mathbf{x}dt = (\mathbf{p}',\mathbf{v}') = (\mathbf{v},\mathbf{a})$
- Need to integrate $\mathbf{x}' = f(\mathbf{x}, t)$




$$\mathbf{x}' = f((\mathbf{p},\mathbf{v}), \mathrm{t}) = (\mathbf{p}',\mathbf{v}')$$

# Phase Space

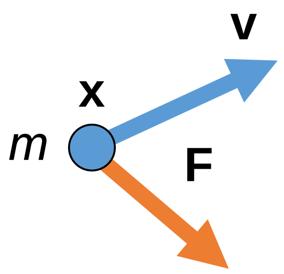
- Position, velocity: $\mathbf{x} = (\mathbf{p},\mathbf{v}) = (px, py, pz, vx, vy, vz)$
- Derivative: $\mathbf{x}' = d\mathbf{x}dt = (\mathbf{p}',\mathbf{v}') = (\mathbf{v},\mathbf{a})$
- Need to integrate $\mathbf{x}' = f(\mathbf{x}, t)$




$$\mathbf{x}' = f((\mathbf{p},\mathbf{v}), \mathrm{t}) = (\mathbf{p}',\mathbf{v}') = (\mathbf{v},\mathbf{a})$$

# Phase Space

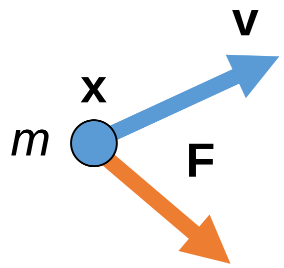
- Position, velocity: **x** = (**p**,**v**) = (*px*, *py*, *pz*, *vx*, *vy*, *vz*)
- Derivative: **x**' = *d***x***dt* = (**p**',**v**') = (**v**,**a**)
- Need to integrate **x**' = *f* (**x**, *t*)




$$\mathbf{x}' = f((\mathbf{p},\mathbf{v}), \mathrm{t}) = (\mathbf{p}',\mathbf{v}') = (\mathbf{v},\mathbf{a}) = (\mathbf{v}, \mathbf{F}/m)$$

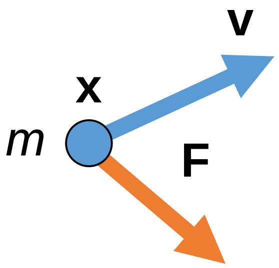
# Data Structures

- Particle: [**p**, **v**, **F**, *m*]
  - **p**: particle position
  - **v**: particle velocity
  - **F**: Force accumulator
  - *m*: particle mass

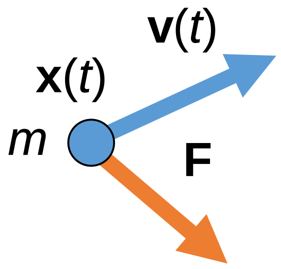


# Data Structures

- Particle: [**p**, **v**, **F**, *m*]
  - **p**: particle position
  - **v**: particle velocity
  - **F**: Force accumulator
  - *m*: particle mass



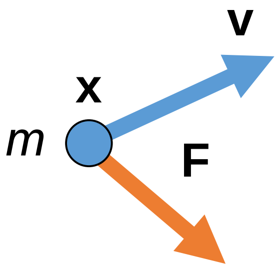

- Solver:
  - State: **x** = (**p**,**v**)
  - Derivative: **x**' = (**v**,**F**/*m*)



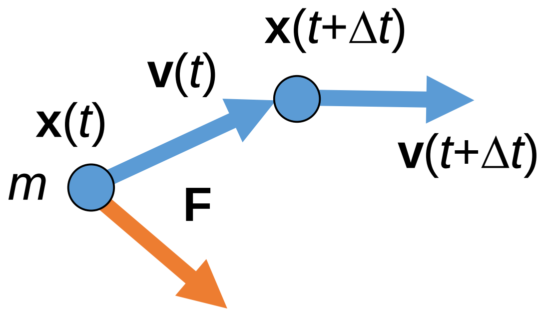

E.g. Euler: $\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \Delta t\, f\,(\mathbf{x}(t),t)$

# Data Structures

- Particle: [**p**, **v**, **F**, $m$]
  - **p**: particle position
  - **v**: particle velocity
  - **F**: Force accumulator
  - $m$: particle mass




- Solver:
  - State: $\mathbf{x} = (\mathbf{p},\mathbf{v})$
  - Derivative: $\mathbf{x}' = (\mathbf{v},\mathbf{F}/m)$

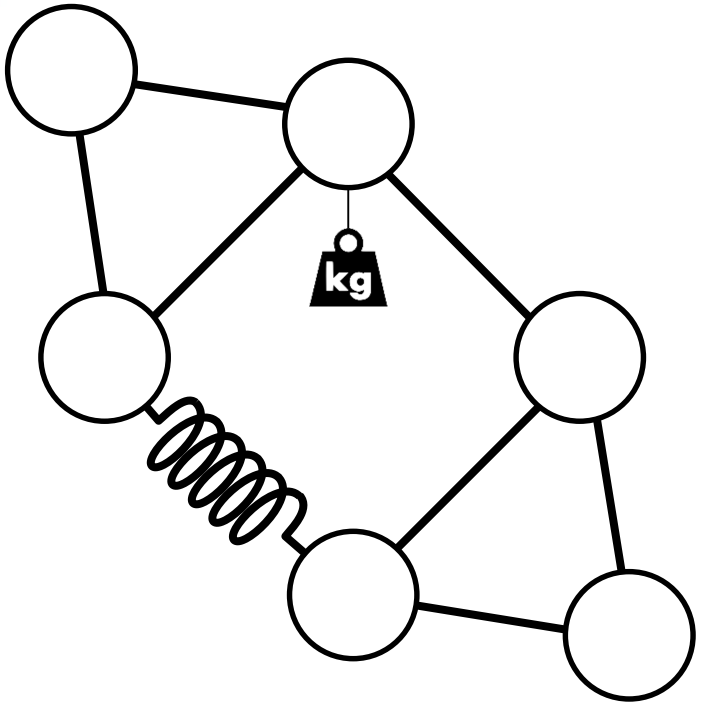



E.g. Euler: $\mathbf{p}(t + \Delta t) \approx \mathbf{p}(t) + \Delta t\, \mathbf{v}(t)$

$\mathbf{v}(t + \Delta t) \approx \mathbf{v}(t) + \Delta t\, \mathbf{F}/m$

# Particle System

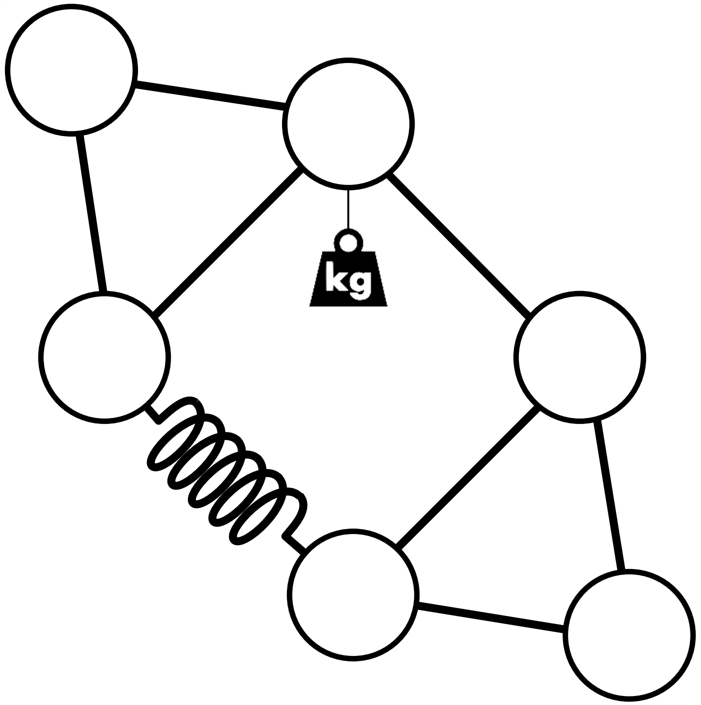
- Particle array: [$\mathbf{p}_i$, $\mathbf{v}_i$, $\mathbf{F}_i$, $m_i$]



# Particle System

- Particle array: [$\mathbf{p}_i$, $\mathbf{v}_i$, $\mathbf{F}_i$, $m_i$]

- Solver:
  - State: $\mathbf{x} = (\mathbf{p}_0, \mathbf{v}_0, \mathbf{p}_1, \mathbf{v}_1, \mathbf{p}_2, \mathbf{v}_2, \ldots)$
  - Derivative: $\mathbf{x}' = (\mathbf{v}_0, \mathbf{F}_0/m_0, \ldots)$

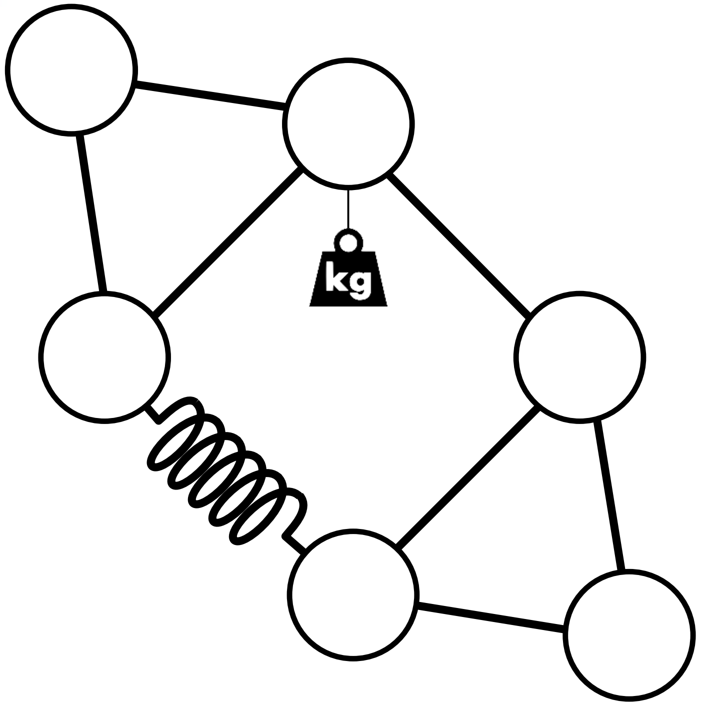


# Particle System

- Particle array: [$\mathbf{p}_i$, $\mathbf{v}_i$, $\mathbf{F}_i$, $m_i$]

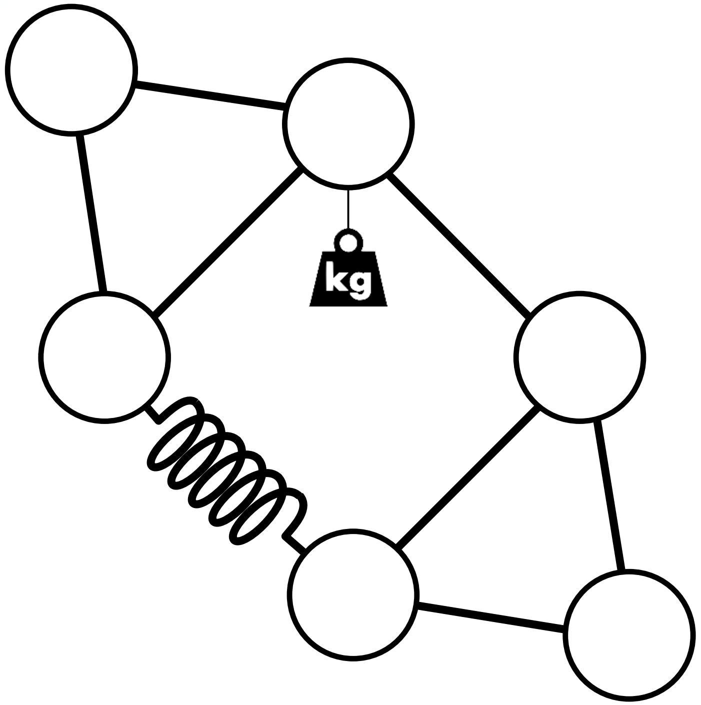
- Solver:
  - State: $\mathbf{x} = (\mathbf{p}_0, \mathbf{v}_0, \mathbf{p}_1, \mathbf{v}_1, \mathbf{p}_2, \mathbf{v}_2, \ldots)$
  - Derivative: $\mathbf{x}' = (\mathbf{v}_0, \mathbf{F}_0/m_0, \ldots)$

- Derivative evaluation
  - For each particle $i$: $\mathbf{F}_i = 0$
  - For each force: add to appropriate $\mathbf{F}_i$
  - For each particle $i$: $\mathbf{p}_i' = \mathbf{v}_i$, $\mathbf{v}_i' = \mathbf{F}_i/m_i$



# Forces

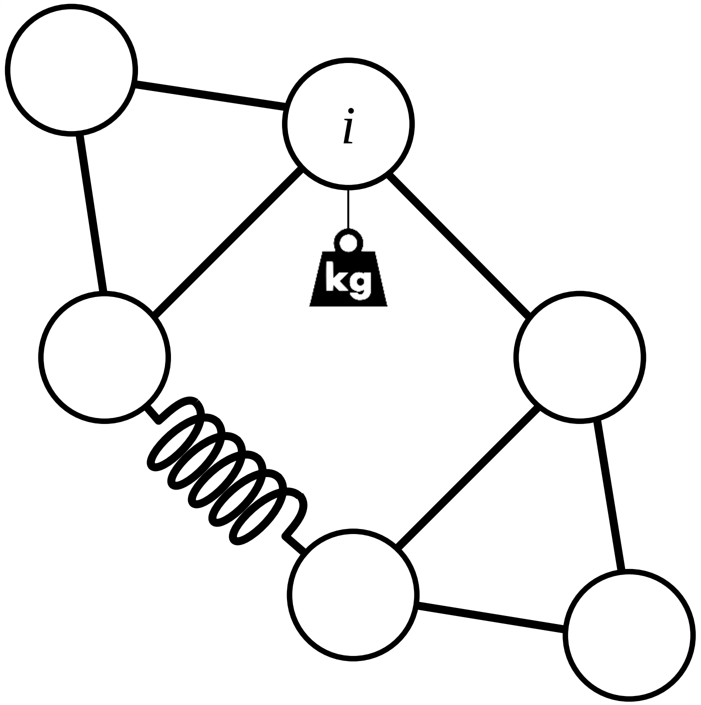
- Derivative evaluation
  - For each particle $i$: $\mathbf{F}_i = 0$
  - For each force: add to appropriate $\mathbf{F}_i$
  - For each particle $i$: $\mathbf{p}_i' = \mathbf{v}_i$, $\mathbf{v}_i' = \mathbf{F}_i/m_i$



# Forces

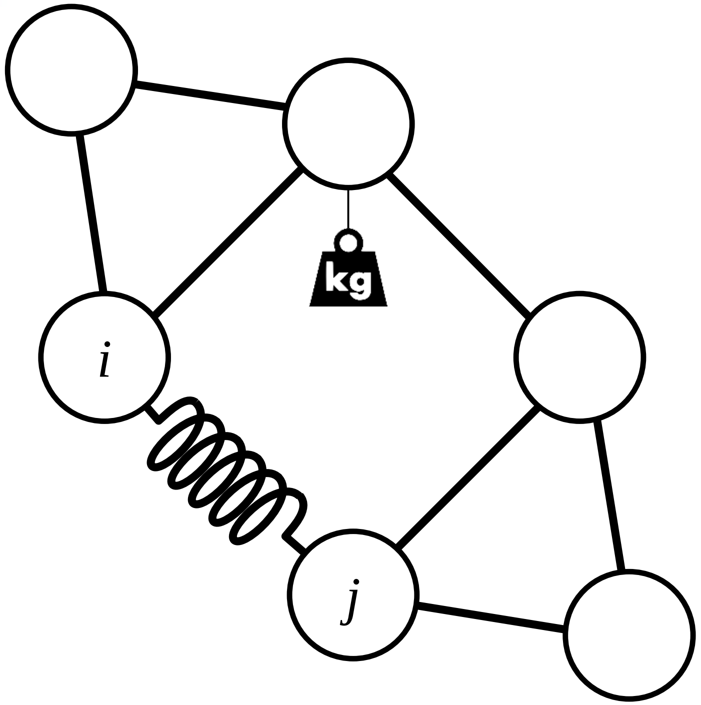
- Derivative evaluation
  - For each particle $i$: $\mathbf{F}_i = 0$
  - For each force: add to appropriate $\mathbf{F}_i$
  - For each particle $i$: $\mathbf{p}_i' = \mathbf{v}_i$, $\mathbf{v}_i' = \mathbf{F}_i/m_i$

- Gravity: $\mathbf{F}_i \mathrel{+}= m_i * \mathbf{G}$



# Forces

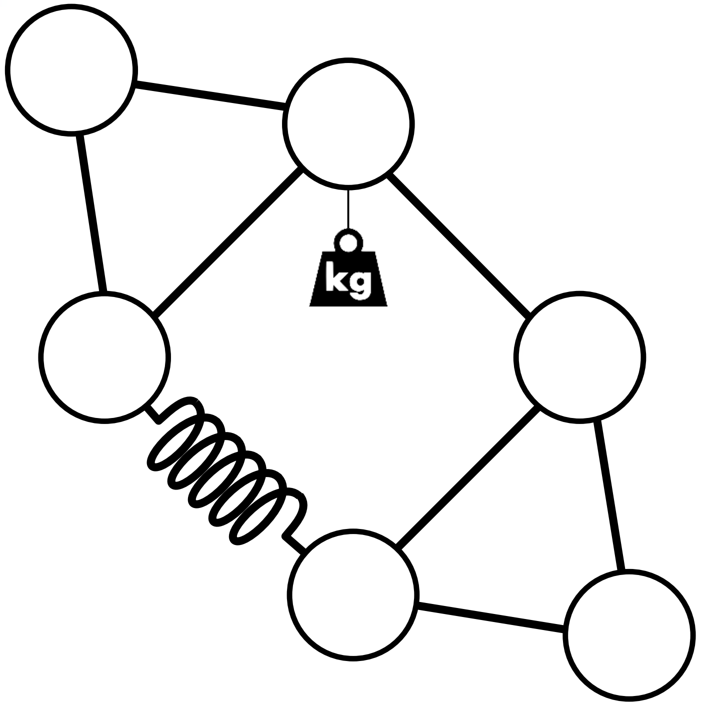
- Derivative evaluation
  - For each particle $i$: $\mathbf{F}_i = 0$
  - For each force: add to appropriate $\mathbf{F}_i$
  - For each particle $i$: $\mathbf{p}_i' = \mathbf{v}_i$, $\mathbf{v}_i' = \mathbf{F}_i/m_i$




- Gravity: $\mathbf{F}_i \mathrel{+}= m_i * \mathbf{G}$
- Spring: $\mathbf{F}_i \mathrel{+}= k*(||\mathbf{p}_j - \mathbf{p}_i|| - r)*(\mathbf{p}_j - \mathbf{p}_i)/||\mathbf{p}_j - \mathbf{p}_i||$
  $\mathbf{F}_j \mathrel{-}=$ same

# Forces

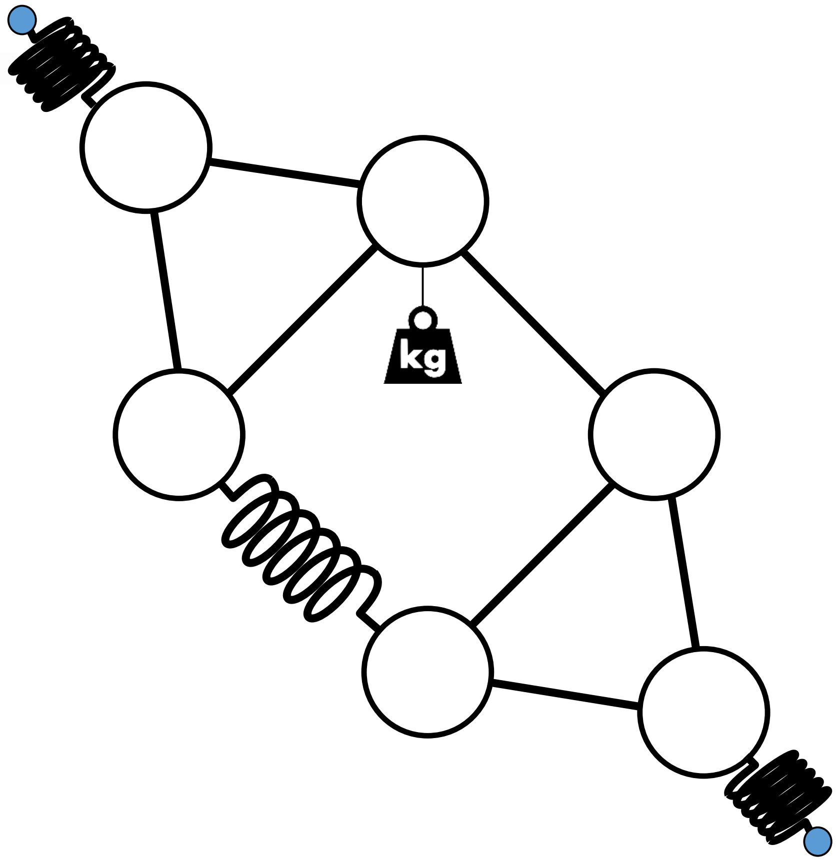
- Derivative evaluation
  - For each particle $i$: $\mathbf{F}_i = 0$
  - For each force: add to appropriate $\mathbf{F}_i$
  - For each particle $i$: $\mathbf{p}_i' = \mathbf{v}_i$, $\mathbf{v}_i' = \mathbf{F}_i/m_i$




- Gravity: $\mathbf{F}_i \mathrel{+}= m_i * \mathbf{G}$
- Spring: $\mathbf{F}_i \mathrel{+}= k*(||\mathbf{p}_j - \mathbf{p}_i|| - r)*(\mathbf{p}_j - \mathbf{p}_i)/||\mathbf{p}_j - \mathbf{p}_i||$
  $\mathbf{F}_j \mathrel{-}=$ same
- Damping: $\mathbf{F}_i \mathrel{-}= k*\mathbf{v}_i$

# Useful Interaction Forces

- Nail
  - Spring between particle and fixed position
  - Force only applied to the one particle at the end of the spring
  - Keeps things from falling off the bottom of the screen



# Useful Interaction Forces

- Nail
  - Spring between particle and fixed position
  - Force only applied to the one particle at the end of the spring
  - Keeps things from falling off the bottom of the screen
- Mouse Spring
  - Spring between particle and mouse position
  - More stable than moving a particle directly to mouse position
  - Clicking the mouse attaches spring with zero rest length to nearest particle