CS**5643**

08 Collision response

Steve Marschner
Cornell University
Spring 2025

Starting simple: particle with fixed obstacle

Reality of collision

- kinetic energy is stored in elastic potential
- energy is released back into kinetic (partly)
- for hard objects this happens very fast

Modeling approximation

- our model doesn't have the DoFs to represent that deformation
- abstract away the details: what is the particle is doing after the collision is over?

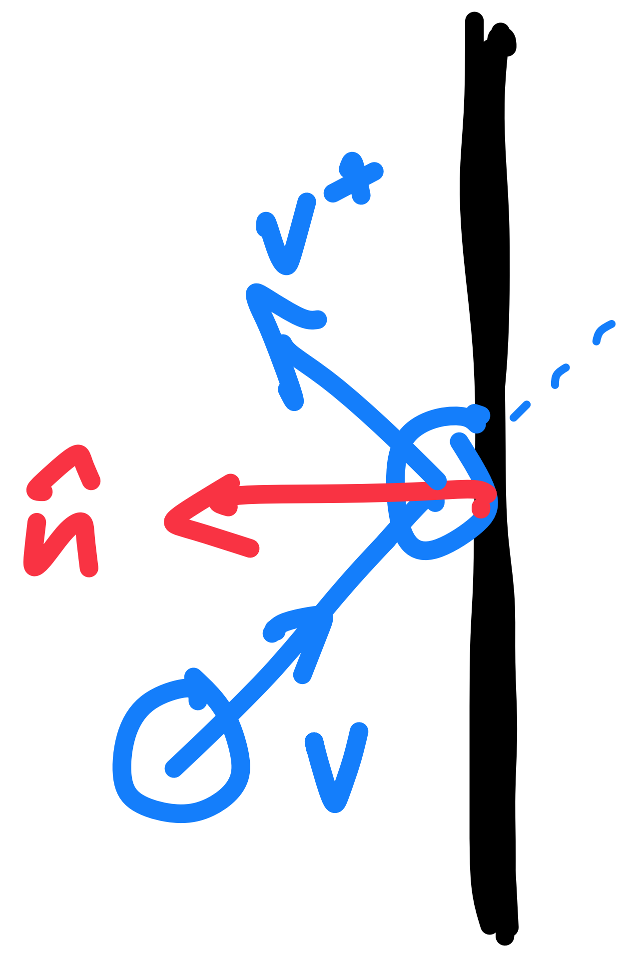
Impluse: summarize force over a short event as a change in momentum

- force applied to ball by wall, and therefore acceleration of ball, varies over a short time
- only final velocity matters: integrate acceleration ($\mathrm{m/s^2}$) over time (s) and forget details
- impulse: integrate force (N) over time (s) to have an analog of force for short events ($\mathrm{N \cdot s}$)

Particle-obstacle collision (frictionless)

Notation: pre-collision velocity $\mathbf{v}$; post-collision $\mathbf{v}^+$

- separate these into normal and tangential: $\mathbf{v} = \mathbf{v}_n + \mathbf{v}_t$; $\mathbf{v}_n = v_n \hat{\mathbf{n}}$; $\mathbf{v}_n \cdot \mathbf{v}_t = 0$
- note $v_n < 0$ otherwise the collision would not be happening



Collision impulse γ acts along contact normal $\hat{\mathbf{n}}$: $\mathbf{v}^+ = \mathbf{v} + \frac{\gamma}{m}\hat{\mathbf{n}}$

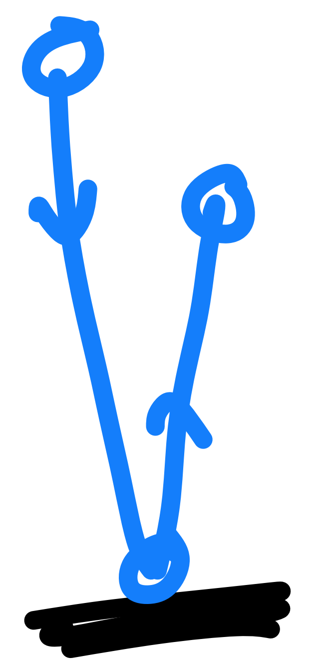
- final velocity is not towards surface, or $v_n^+ \geq 0$; $\mathbf{v}_t^+ = \mathbf{v}_t$

Decide magnitude of impulse by conservation of energy

- $E_k^{\text{before}} = \frac{1}{2}m\mathbf{v}^2 = \frac{1}{2}m(\mathbf{v}_n^2 + \mathbf{v}_t^2) = \frac{1}{2}m\mathbf{v}_t^2 + \frac{1}{2}mv_n^2$
- $E_k^{\text{after}} = \frac{1}{2}m(\mathbf{v}^+)^2 = \frac{1}{2}m((\mathbf{v}_n^+)^2 + \mathbf{v}_t^2) = \frac{1}{2}m\mathbf{v}_t^2 + \frac{1}{2}m(v_n^+)^2$
- so normal component has to exactly reverse to conserve energy: $v_n^+ = -v_n$, so $\gamma = -2mv_n$

Particle-obstacle collision (with losses)

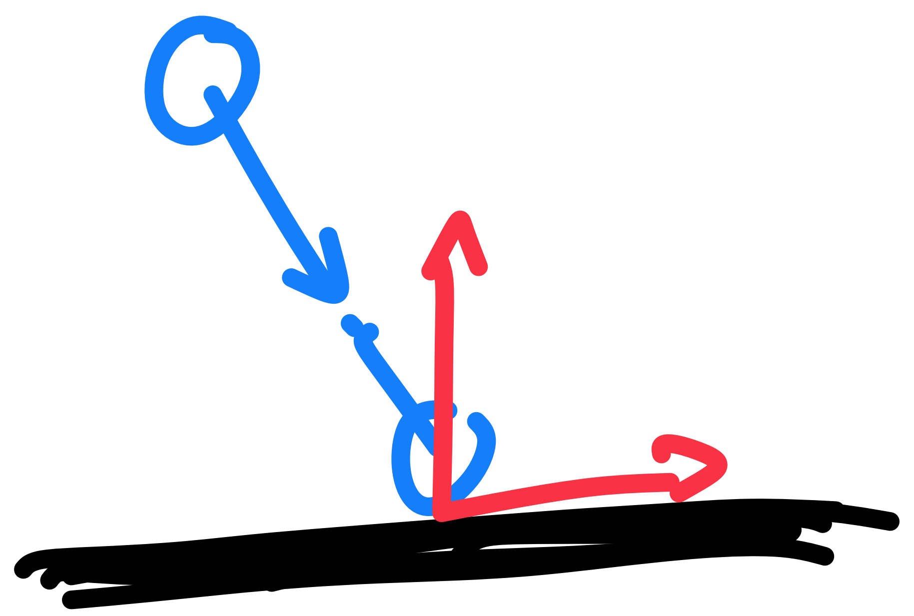
Restitution



- model energy loss with heuristic "coefficient of restitution" c_r such that $v_n^+ = -c_r v_n$;
  $\gamma = -(1 + c_r) m v_n$

Friction impulse γ_f acts in the tangential direction

- Coulomb friction model: frictional force $f_f \leq \mu f_n$
- while there is tangential velocity $f_f = \mu f_n$, integrates to $\gamma_f = \mu \gamma_n$
- friction does not take tangential velocity past zero so $\gamma_f = \min(\mu \gamma_n, m v_t)$
- $v_t^+ = v_t - \frac{\gamma_f}{m}$ and $\mathbf{v}_t^+ = v_t^+ \hat{\mathbf{v}}_t$



Elastic collision between particles in 1D

Momentum conservation: apply opposite impulses Δp and $-\Delta p$

- after applying collision impulse $\dot{x}^+ = \dot{x} + \frac{\Delta p}{m_x}$ and $\dot{y}^+ = \dot{y} - \frac{\Delta p}{m_y}$

Energy conservation ensured by reversing the relative velocity

- kinetic energy before collision: $E_k^{\text{before}} = \frac{1}{2}(m_x \dot{x}^2 + m_y \dot{y}^2)$
- energy after collision: $E_k^{\text{after}} = \frac{1}{2}\left(m_x(\dot{x} + \frac{\Delta p}{m_x})^2 + m_y(\dot{y} - \frac{\Delta p}{m_y})^2 \right)$
- $E_k^{\text{after}} - E_k^{\text{before}} = (\dot{x} - \dot{y})\Delta p + \frac{\Delta p^2}{2m_x} + \frac{\Delta p^2}{2m_y} = v_{\text{rel}}\Delta p + \frac{1}{2}\left(\frac{1}{m_x} + \frac{1}{m_y}\right)\Delta p^2$
- Set change to zero $\Longrightarrow \Delta p = 0$ or $\Delta p = -2m_{\text{eff}}v_{\text{rel}}$

  where $v_{\text{rel}} = \dot{x} - \dot{y}$ and $m_{\text{eff}} = \left(\frac{1}{m_x} + \frac{1}{m_y}\right)^{-1}$

Collision response for elastic colliding particles

Collision impulse acts along the collision normal

- use of an impulse ensures momentum conservation
- $m_x \Delta \dot{\mathbf{x}} = - m_y \Delta \dot{\mathbf{y}} = \Delta \mathbf{p}$

To compute impulse, separate into normal and tangential components

- $\dot{\mathbf{x}} = \dot{\mathbf{x}}_n + \dot{\mathbf{x}}_t$ and $\dot{\mathbf{y}} = \dot{\mathbf{y}}_n + \dot{\mathbf{y}}_t$; kinetic energy of $\mathbf{x}$ is $\frac{1}{2} m_x \dot{\mathbf{x}}_n^2 + \frac{1}{2} m_x \dot{\mathbf{x}}_t^2$ and similar for $\mathbf{y}$
- normal impulse only affects the normal part of the energy, so conserve that
- …but this is the same 1D problem again!
- $\Delta \mathbf{p} = \gamma \hat{\mathbf{n}}; \gamma = = - 2 m_{\text{eff}} v_n$
  - where $v_n = \hat{\mathbf{n}} \cdot (\dot{\mathbf{x}} - \dot{\mathbf{y}})$ is the normal component of the relative velocity

Restitution and friction in two-particle case

We've seen that conserving energy in a two-particle collision translates to exactly reversing the relative normal velocity

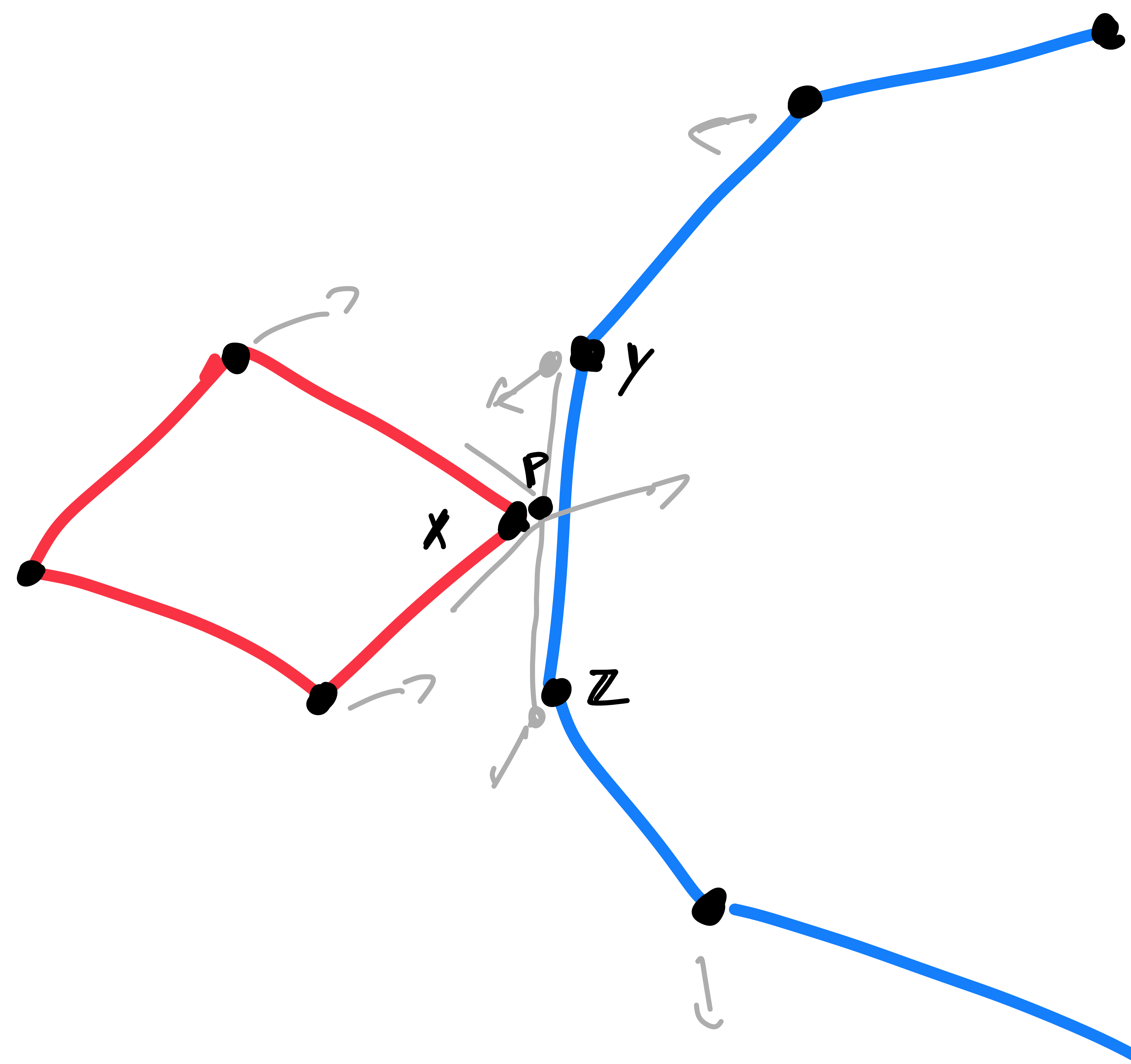
- this was the same as in the solid-wall collision
- we can compute normal and friction impulses using the same approach
- scale down normal impulse: $\Delta \mathbf{p} = \gamma \hat{\mathbf{n}}; \gamma = -(1 + c_r) m_{\text{eff}} v_n$ where c_r is the coeff. of restitution
- friction impulse acts along the tangential component of *relative* velocity
  - still a fraction of the normal impulse
  - still limited to zeroing out the tangential relative velocity
  - $\gamma_f = -\min(\mu\gamma, m v_t); \Delta \mathbf{p} = \gamma_f \hat{\mathbf{v}}_t$

Collisions with deformables involving edges in 2D

In 2D remember that vertex-edge collisions are the ones we worry about

To resolve a collision we need to apply impulses to three vertices

- contact is between the moving vertex and a point on the moving edge
  - moving point $\mathbf{x}$; edge vertices $\mathbf{y}$ and $\mathbf{z}$
  - colliding point $\mathbf{p} = \alpha\mathbf{y} + \beta\mathbf{z}$ where $\alpha + \beta = 1$
- impulses are designed to achieve the desired change in relative velocity between $\mathbf{x}$ and $\mathbf{p}$
- to derive required impulse, need to decide how the impulse will be distributed between ends
  - typical: barycentric weighting
  - $\gamma_x = \gamma; \gamma y = -\alpha\gamma; \gamma_z = -\beta\gamma$



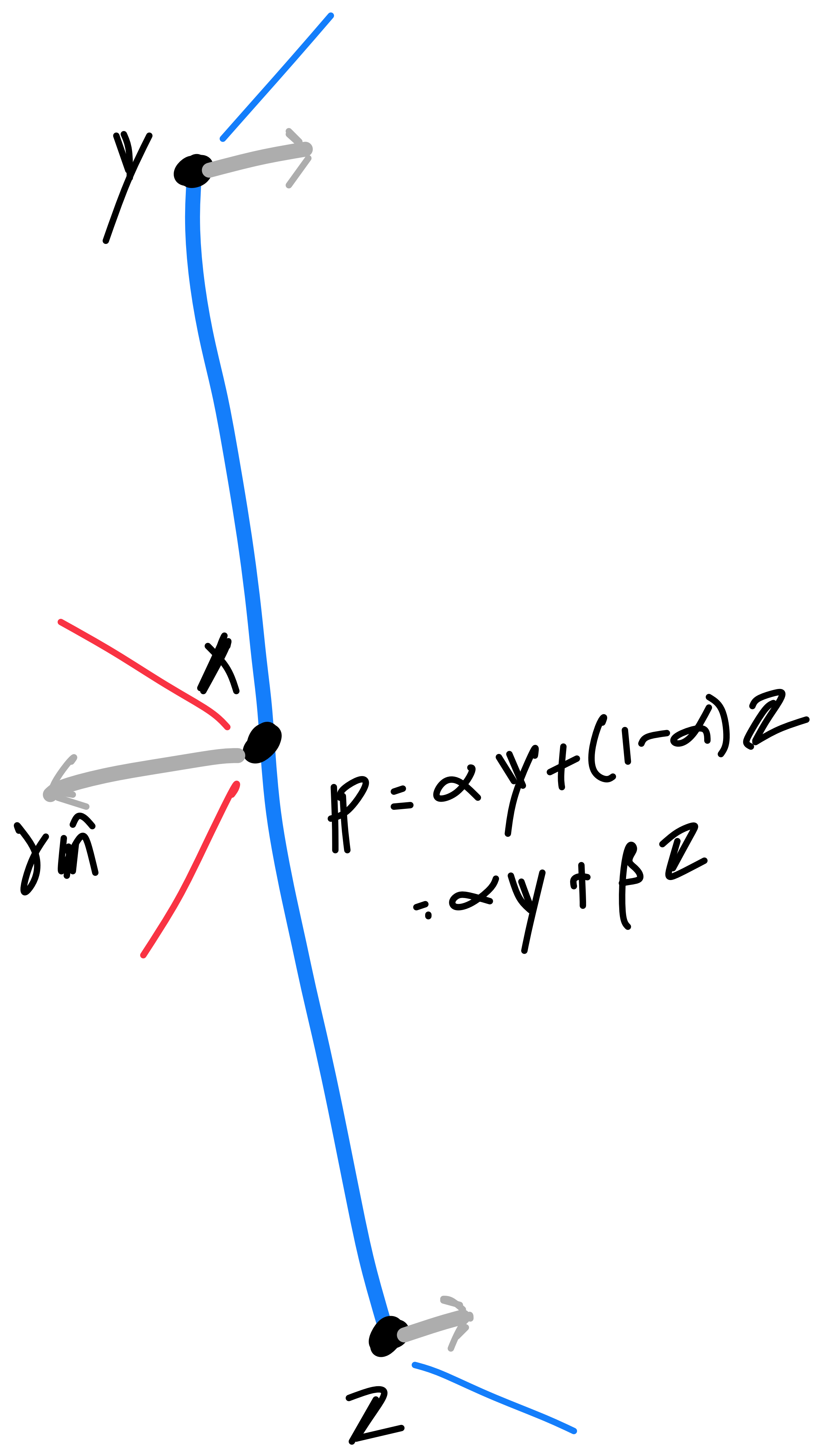
Collisions with deformables involving edges in 2D

Positions:

- $\mathbf{x}(t) = \mathbf{x} + t\dot{\mathbf{x}}$; $\mathbf{y}(t) = \mathbf{y} + t\dot{\mathbf{y}}$; $\mathbf{z}(t) = \mathbf{z} + t\dot{\mathbf{z}}$
- $\mathbf{p}(t) = \mathbf{p} + \alpha t\dot{\mathbf{y}} + \beta t\dot{\mathbf{z}}$; $\dot{\mathbf{p}} = \alpha\dot{\mathbf{y}} + \beta\dot{\mathbf{z}}$
- $\mathbf{x}(t_c) = \mathbf{p}(t_c)$

Post-collision velocities:

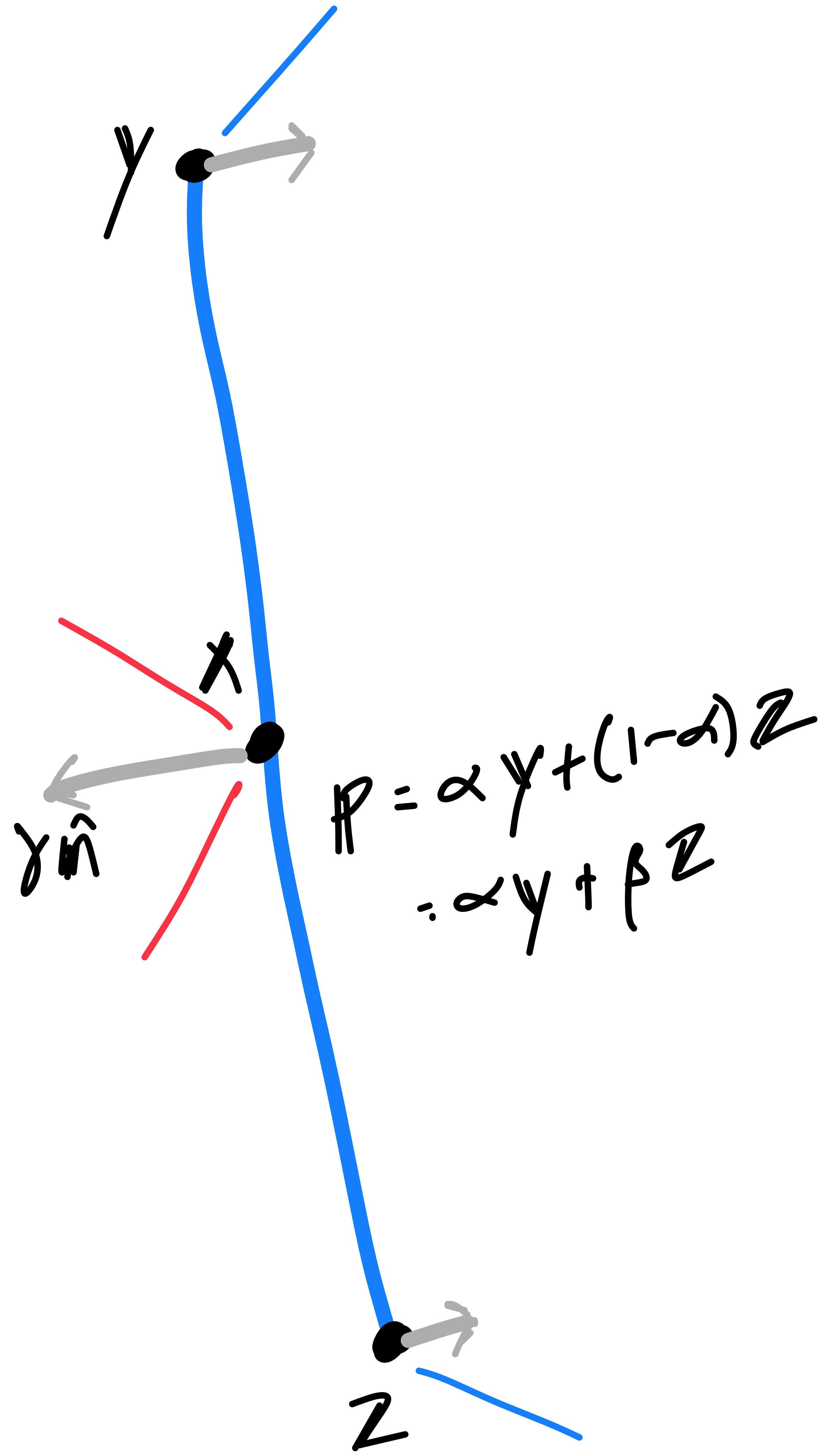
- $\dot{\mathbf{x}}^+ = \dot{\mathbf{x}} + \frac{\gamma}{m_x}\hat{\mathbf{n}}$; $\dot{\mathbf{y}}^+ = \dot{\mathbf{y}} - \frac{\alpha\gamma}{m_y}\hat{\mathbf{n}}$; $\dot{\mathbf{z}}^+ = \dot{\mathbf{z}} - \frac{\beta\gamma}{m_z}\hat{\mathbf{n}}$
- $\dot{\mathbf{p}}^+ = \alpha\dot{\mathbf{y}}^+ + \beta\dot{\mathbf{z}}^+$



Collisions with deformables involving edges in 2D

Normal components:

- $\dot{x}_n^+ = \dot{x}_n + \frac{\gamma}{m_x}$; $\dot{p}_n^+ = \dot{p}_n - \frac{\alpha^2\gamma}{m_y} - \frac{\beta^2\gamma}{m_z}$
- $v_n = \dot{x}_n - \dot{p}_n$
- $v_n^+ = \dot{x}_n^+ - \dot{p}_n^+ = -c_r v_n$
- $-c_r v_n = \dot{x}_n - \dot{p}_n + \left(\frac{1}{m_x} + \frac{\alpha^2}{m_y} + \frac{\beta^2}{m_z}\right)\gamma$
- $(1 + c_r)v_n = -\left(\frac{1}{m_x} + \frac{\alpha^2}{m_y} + \frac{\beta^2}{m_z}\right)\gamma$
- $\gamma = -(1 + c_r)m_{\text{eff}} v_n$



Collisions with deformables involving edges in 2D

Positions:

- $\mathbf{x}(t) = \mathbf{x} + t\dot{\mathbf{x}}$; $\mathbf{y}(t) = \mathbf{y} + t\dot{\mathbf{y}}$;
  $\mathbf{z}(t) = \mathbf{z} + t\dot{\mathbf{z}}$
- $\mathbf{p}(t) = \mathbf{p} + \alpha t\dot{\mathbf{y}} + \beta t\dot{\mathbf{z}}$; $\dot{\mathbf{p}} = \alpha\dot{\mathbf{y}} + \beta\dot{\mathbf{z}}$
- $\mathbf{x}(t_c) = \mathbf{p}(t_c)$

Post-collision velocities:

- $\dot{\mathbf{x}}^+ = \dot{\mathbf{x}} + \frac{\gamma}{m_x}\hat{\mathbf{n}}$; $\dot{\mathbf{y}}^+ = \dot{\mathbf{y}} - \frac{\alpha\gamma}{m_y}\hat{\mathbf{n}}$;
  $\dot{\mathbf{z}}^+ = \dot{\mathbf{z}} - \frac{\beta\gamma}{m_z}\hat{\mathbf{n}}$
- $\dot{\mathbf{p}}^+ = \alpha\dot{\mathbf{x}} + \beta\dot{\mathbf{y}}$

Normal components:

- $\dot{x}_n^+ = \dot{x}_n + \frac{\gamma}{m_x}$; $\dot{p}_n^+ = \dot{p}_n - \frac{\alpha\gamma}{m_y} - \frac{\beta\gamma}{m_z}$
- $v_n = \dot{x}_n - \dot{p}_n$
- $v_n^+ = \dot{x}_n^+ - \dot{p}_n^+ = -c_r v_n$
- $-c_r v_n = \dot{x}_n - \dot{p}_n + \left(\frac{1}{m_x} + \frac{\alpha}{m_y} + \frac{\beta}{m_z}\right)\gamma$
- $(1 + c_r)v_n = -\left(\frac{1}{m_x} + \frac{\alpha}{m_y} + \frac{\beta}{m_z}\right)\gamma$
- $\gamma = -(1 + c_r)m_{\text{eff}}v_n$

Resolving multiple collisions

This is where it gets messy!

Resolving collisions one at a time can work in easy cases

- when there are not too many collisions
- when the collisions are generally well separated in time
- when there is no resting contact

In harder cases collisions are highly interdependent

- consider a stack of 5 boxes…
- adding friction makes things even worse
- collision problems can even encode NP-hard problems, in theory

Result: large variety of collision response algorithms, few ironclad guarantees

Broad map of collision methods

Penalties and barriers

- devise forces that vary smoothly and push objects apart
- older idea: penalty forces that activate when objects interpenetrate
- newer idea: barrier potentials that activate on proximity and prevent interpenetration
- the good: smoothly varying forces, fewer discrete decisions to make
- the bad: forces have to be very stiff to be effective, leading to integration challenges

Broad map of collision methods

Impulses

- instantaneous events that happen exactly at the time of collision
- really simple way to handle well separated collisions
- computing impulses separately doesn’t always handle simultaneous collisions
- the good: impulses don’t add stiffness, can be simple and fast
- the bad: no principled handling of simultaneous collisions

Broad map of collision methods

Constraints

- consider many simultaneous collisions as constraints on motion
- solve a system of equations to find a simultaneous solution to all constraints
- many solution methods, from heavy global solvers to simple iterations
- iterative solvers look a lot like resolving contacts separately
- the good: doesn't add stiffness, can solve complex cases
- the bad: methods can be complex, hard to guarantee robustness in all situations

Broad map of collision methods

Strategies for resolving collisions

- recall the Symplectic Euler integrator
    - 1. compute acceleration $\mathbf{a}_0 = M^{-1}\mathbf{f}(t_0)$
    - 2. compute velocity $\mathbf{v}_1 = \mathbf{v}_0 + h\mathbf{a}_0$
    - 3. compute position $\mathbf{x}_1 = \mathbf{x}_0 + h\mathbf{v}_1$
- "acceleration level" methods think about forces and accelerations and make changes at step 1
- "velocity level" methods think about impulses and velocities and make chances after step 2
- "position level" methods think about correcting positions directly and make changes after step 3

Choice of collision method

Depends on type of simulation

- deformables have many contacts but more local interactions
- rigid bodies have more global interactions (more on that later)
- solids can recover from interpenetration; thin objects (rods, sheets) can't

Collision response choice

- for robustness and accuracy with extreme deformations, barrier potentials
- for efficiency with rigid bodies or stiff solids, impulses or iterative constraint solvers
- for accuracy, global constraint solvers (becoming less used)

Collision detection choice

- for solids and rigid bodies, often instantaneous overlap query
- for cloth and rods, often continuous collision detection
- if using barrier potentials, proximity queries

Simple method #1: sequential resolution

Strategy: simulate to the first collision, fix it, then continue

- assume Symplectic Euler, first updating velocities then positions
- 1. compute forces $\mathbf{f}_0$ at the start of the step, t_0, set $t = t_0$
- 2. compute new velocities $\mathbf{v}_1 = \mathbf{v}_0 + hM^{-1}\mathbf{f}_0$
- 3. perform CCD over $[t, t_1]$ to find any collisions
    - if there are any collisions, find the one that happens first, call that time t_c
    - advance all positions to time $t = t_c$
    - compute an impulse to resolve the collision, update the directly involved velocities
    - repeat this step until there are no more collisions
- 4. advance all positions from t to t_1

Simple method #2: parallel resolution

Strategy: fix all collisions in the timestep, then check if we broke anything

- assume Symplectic Euler, first updating velocities then positions
- 1. compute forces $\mathbf{f}_0$ at the start of the step, t_0, set $t = t_0$
- 2. compute new velocities $\mathbf{v}_1 = \mathbf{v}_0 + hM^{-1}\mathbf{f}_0$
- 3. perform CCD over $[t_0, t_1]$ to find any collisions
  - order collisions by their times
  - for each collision:
    - compute an impulse using vertex positions at the collision time
    - apply the impulse to update the directly involved velocities
  - after resolving all collisions, repeat this whole step until there are no more collisions
- 4. advance all positions from t_0 to t_1