XPBD: position-based simulation of compliant constrained dynamics

IG3DA/IMA904 : Advanced 3D Computer Graphics

Sammy Rasamimanana

February 10, 2023

Table of Contents

Introduction

A basic strategy for physic simulation

Consider a particle with

1. mass m
2. position p
3. speed v

Newton's Second Law of Motion

$$m\dot{v} = F_{ext}$$

$$\dot{p} = v$$

Semi-implicit Euler : a basic strategy for physic simulation

Let's make a little change

Semi-implicit Euler/Symplectic integrator

Let dt the time step and t_k the time at the k-th step of the animation. Thus

$$v_{k+1} = v_k + \frac{F_{ext}}{m} dt$$

$$p_{k+1} = p_k + v_{k+1} dt$$

Algorithm

Algorithm 1 Simulation loop

1: **for** all particles **do**
2: $v \longleftarrow v + f_{ext} \cdot m^{-1} \cdot dt$
3: $p \longleftarrow p + v \cdot dt$
4: **end for**

Position Based Dynamics (PBD)

Another approach: a constraints based strategy

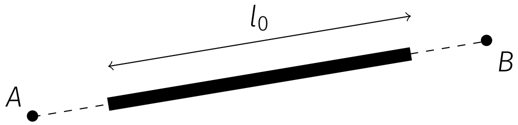
A dynamic object : a set of N vertices (= particles) and M constraints

A Constraint $j \in [\![1, M]\!]$ consists of

- a cardinality n_j and a set of indices $i_1, \cdots, i_{n_j} \subset [\![1, N]\!]$
- a function $C_j : \mathbb{R}^{3n_j} \longrightarrow \mathbb{R}$
- a stiffness parameter $k_j \in [0, 1]$
- a type of either *equality* ($C_j = 0$) or *inequality* ($C_j \geqslant 0$)

Constraint examples

Two particles A and B.



Distance constraint : $C(p_A, p_B) = |p_A - p_B| - l_0$

It is an *equality* constraint : satisfied when $C(p_A, p_B) = 0$.

Main idea : projecting the points

Move the points such that they satisfy the constraint

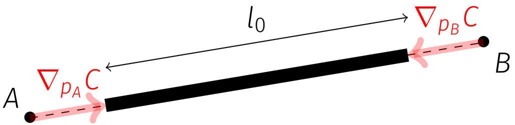
Main idea : move along gradient

Two particles A and B with Distance constraint : $C(p_A, p_B) = |p_A - p_B| - l_0$

Denote $p = \begin{pmatrix} p_A^T \\ p_B^T \end{pmatrix} = \begin{pmatrix} x_A & y_A & z_A \\ x_B & y_B & z_B \end{pmatrix}$. How to find Δp such that $C(p + \Delta p) = 0$?

Gradient $\nabla_p C$ indicates the direction of maximal increase for C

The correction term Δp will be along the gradient $\nabla_p C$



PBD Algorithm

Algorithm 2 Simulation loop

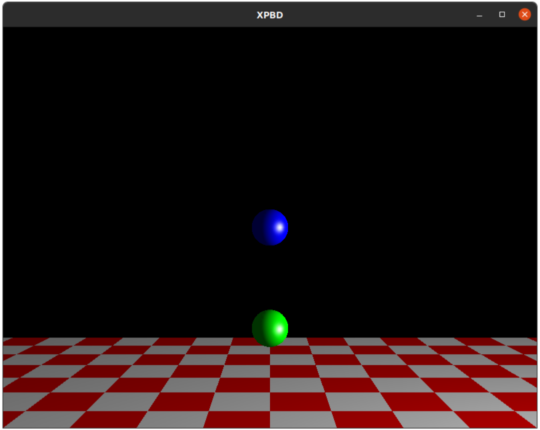
1: **for** all particles **do**
2: $p_{old} \longleftarrow p$
3: $v \longleftarrow v + f_{ext} \cdot m^{-1} \cdot dt$
4: $p \longleftarrow p + v \cdot dt$
5: **end for**
6: **while** $i <$ ***nbIterations*** **do**
7: **for** all constraints C **do**
8: $\Delta p \longleftarrow$ *solveConstraint(C)*
9: $p \longleftarrow p + \Delta p$
10: **end for**
11: $i \longleftarrow i + 1$
12: **end while**
13: $v \longleftarrow (p - p_{old})/dt$

Roadmap

Goals of the project

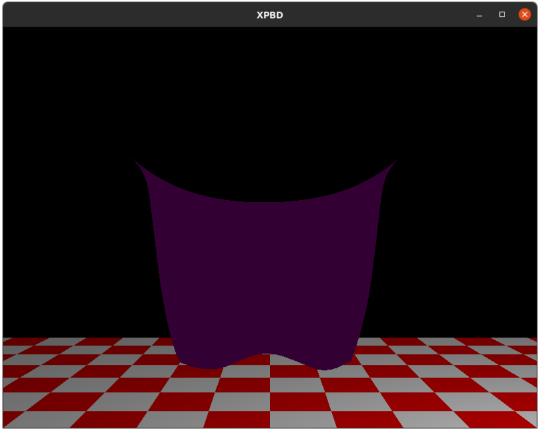
- Implementation of distance constraint and fixed constraint (often used)
- Tests on a spring model and on a cloth
- Implementation of bending constraint and collision/penetration constraints
- Tests cloth and ball interaction with ground

Scene: Spring



Two particles with a distance constraint

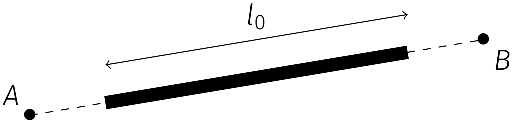
Scene: Cloth simulation (1)



20×20 particles with distance constraints for each edge of the quads

Collision constraint

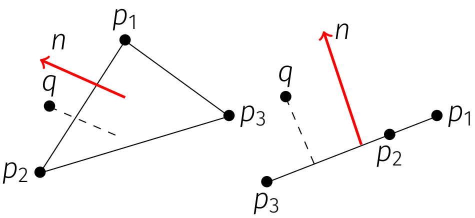
Two particles A and B.



Distance constraint : $C(p_A, p_B) = |p_A - p_B| - l_0$

But its *inequality* version: satisfied when $C(p_A, p_B) \geqslant 0$.

Penetration Constraint

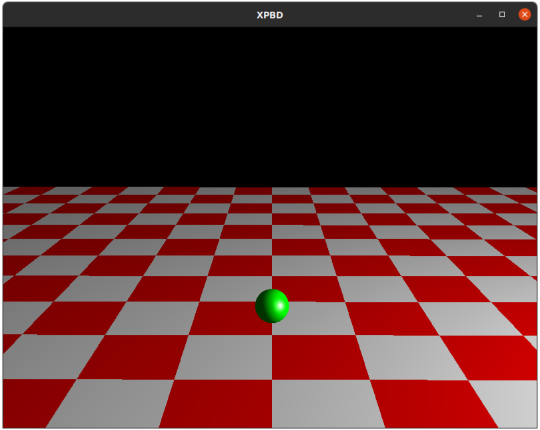


$$C = (q - p_1) \cdot n \geqslant 0 \text{ where } n = \frac{(p_2 - p_1) \times (p_3 - p_1)}{||(p_2 - p_1) \times (p_3 - p_1)||}$$

Useful to model collision with walls or floor

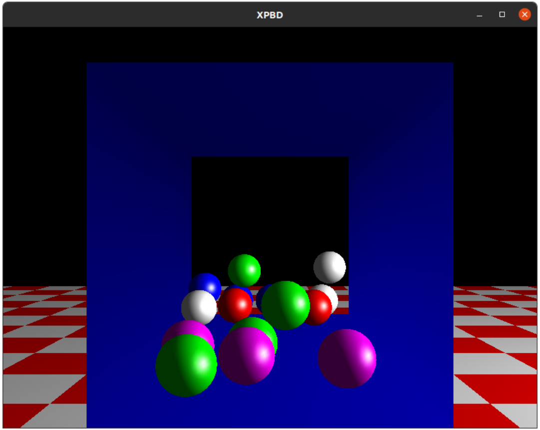
If p_1, p_2, p_3 are points of the wall, they are constants and C is just a function of q.

Scene: Floor collision



One particle with two wall-penetration constraints

Scene: Balls

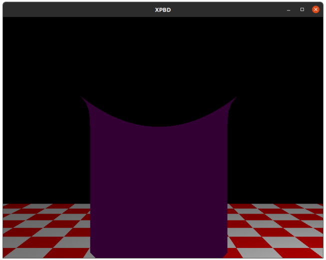


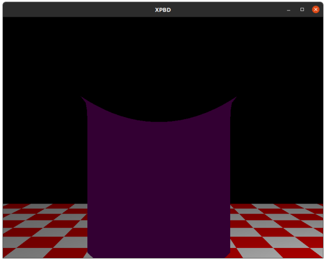
Collision between particles + Wall constraints

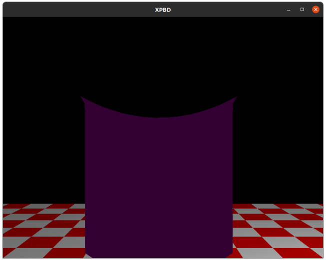
PBD : pros and cons

- ➕ direct manipulation of positions of vertices and parts of objects
- ➕ gives control over explicit integration and removes typical instability problems (it is unconditionnally stable)
- ➕ can handle non-linear constraints
- ➖ low accuracy, no energy preserving
- ➖ stiffness of the model depends on the number of sub-iterations and the time step when moving along the gradient
- ➖ results depends on a lot of parameters: order of constraints, stiffness, mass, etc. and do not have a proper physical interpretation

Scene: Cloth simulation (2)







Stiffness dependance of PBD

Stiffness of the model depends on the number of sub-iterations and the time step when moving along the gradient

Extended Position Based Dynamics (XPBD)

Compliant constraint formulation

Idea 1: treat a constraint as an energy potential

$$U(p) = \frac{1}{2} C(p)^T \alpha^{-1} C(p)$$

α corresponds to the inverse stiffness

Newton's second law : $M\ddot{p} = -\nabla_p U^T = -\nabla_p C^T(p) \alpha^{-1} C(p)$

Compliant constraint formulation

Newton's second law : $M\ddot{p} = -\nabla_p U^T = -\nabla_p C^T(p)\alpha^{-1}C(p)$

Idea 2: introduce the Lagrange multiplier

$$\lambda = -\tilde{\alpha}^{-1}C(p)$$

The equation becomes the system

$$\begin{aligned} M\ddot{p} - \nabla_p C(p)^T\lambda &= 0 \\ C(p) + \tilde{\alpha}\lambda &= 0 \end{aligned}$$

$\Longrightarrow$ Discretization of the system and iterative resolution (Gauss-Seidel or Jacobi) to find λ then p at each time.

XPBD

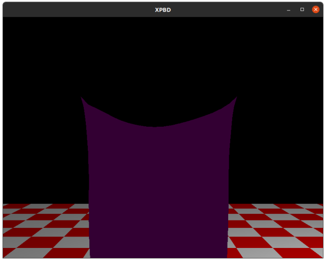
- direct manipulation of positions of vertices and parts of objects
- gives control over explicit integration and removes typical instability problems (it is unconditionnally stable)
- can handle non-linear constraints
- no energy preserving
- stiffness of the model qualitatively independent of time step and number of iterations
- results depends on a lot of parameters: order of constraints, stiffness, mass, etc. and do not have a proper physical interpretation

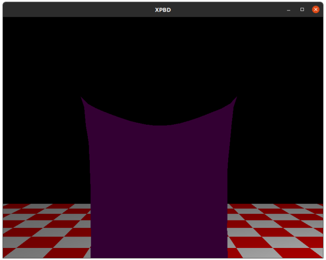
XPBD Algorithm

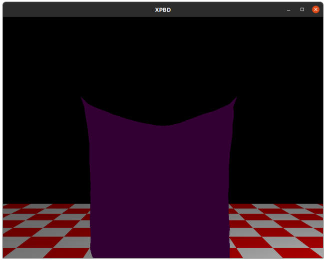
Algorithm 3 Simulation loop

for all particles **do**
  $p_{old} \longleftarrow p$
  $v \longleftarrow v + f_{ext} \cdot m^{-1} \cdot dt$
  $p \longleftarrow p + v \cdot dt$
end for
Initialize multipliers $\lambda \longleftarrow 0$
while $i < \texttt{nbIterations}$ **do**
  for all constraints C **do**
    Compute $\Delta\lambda$
    Compute Δp
    $\lambda \longleftarrow \lambda + \Delta\lambda$
    $p \longleftarrow p + \Delta p$
  end for
  $i \longleftarrow i + 1$
end while
$v \longleftarrow (p - p_{old})/dt$

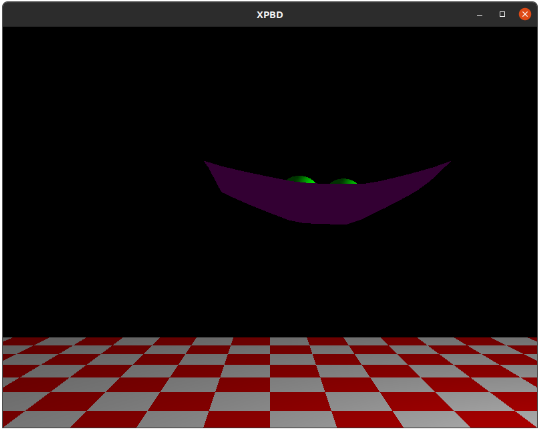
Scene: Cloth simulation (3)







Scene: Cloth balloon



Conclusion

Goals of the project

- ✓ Implementation of distance constraint and fixed constraint (often used)
- ✓ Tests on a spring model and on a cloth
- ✓ Implementation of bending constraint and collision/penetration constraints
- ✓ Tests cloth and ball interaction with ground
- Hierarchical or parallelized constraints implementation
- Implementation of Rayleigh dissipation potential (for damping)

Thank You

Questions?

Still some unstability

Semi-implicit Euler/Symplectic integrator

Let dt the time step and t_k the time at the k-th step of the animation. Thus

$$v_{k+1} = v_k + \frac{F_{ext}}{m} dt$$

$$p_{k+1} = p_k + v_{k+1} dt$$

$\Longrightarrow$ More stable than explicit Euler, but still unstable depending on time step and types of force ...

$\Longrightarrow$ A dynamic object : its center of mass (usually)

Force based strategy for physic simulation

Consider a particle with

1. mass m
2. position p
3. speed v

Explicit Euler: discretization of Newton's Second Law of Motion

Let dt the time step and t_k the time at the k-th step of the animation. Thus

$$m(v_{k+1} - v_k) = F_{ext} dt$$

$$p_{k+1} - p_k = v_k dt$$

Explicit Euler : a basic strategy for physic simulation

Consider a particle with mass m, position p, speed v

Explicit Euler: discretization of Newton's Second Law of Motion

Let dt the time step and t_k the time at the k-th step of the animation. Thus

$$v_{k+1} = v_k + \frac{F_{ext}}{m} dt$$

$$p_{k+1} = p_k + v_k dt$$

Explicit Euler : a basic strategy for physic simulation

Consider a particle with mass m, position p, speed v

Explicit Euler: discretization of Newton's Second Law of Motion

Let dt the time step and t_k the time at the k-th step of the animation. Thus

$$v_{k+1} = v_k + \frac{F_{ext}}{m} dt$$

$$p_{k+1} = p_k + v_k dt$$

But it diverges and has a low accuracy !