

# XPBD: position-based simulation of compliant constrained dynamics

IG3DA/IMA904 : Advanced 3D Computer Graphics

---

Sammy Rasamimanana

November 25, 2022

# Table of Contents

1. Introduction
2. Position Based Dynamics (PBD)
3. Extended Position Based Dynamics (XPBD)
4. Conclusion

# 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

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

$$\dot{p} = v$$

# 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 !

# 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 1 Simulation loop

---

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

---

# Position Based Dynamics (PBD)

---

### Semi-implicit Euler/Symplectic integrator

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

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

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

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

⇒ A dynamic object : its center of mass (usually)

## Another approach: a constraints based strategy

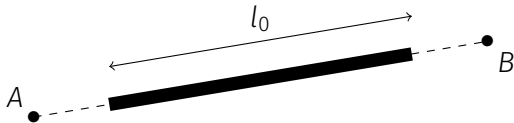
A dynamic object : a set of  $N$  vertices (= particles) and  $M$  constraints

**i** A Constraint  $j \in \llbracket 1, M \rrbracket$  consists of

- a cardinality  $n_j$  and a set of indices  $i_1, \dots, i_{n_j} \subset \llbracket 1, N \rrbracket$
- 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* ( $C_j \geq 0$ )

# 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

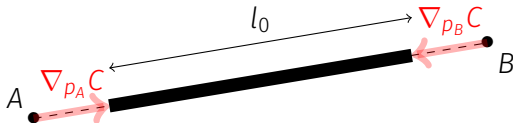
## Main idea : move along gradient

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  $\Delta p$  such that  $C(p + \Delta p) = 0$  ?

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

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



---

## Algorithm 2 Simulation loop

---

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

---

## 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 timpe step when moving along the gradient



# Extended Position Based Dynamics (XPBD)

---

💡 Idea 1: treat a constraint as an energy potential

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

$\alpha$  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

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

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

- ⊕ direct manipulation of positions of vertices and parts of objects
- ⊕ gives control over explicit integration and removes typical instability problems (it is unconditionally stable)
- ⊕ can handle non-linear constraints
- ⊖ no energy preserving
- ⊕ stiffness of the model qualitatively independent of time step and number of iterations

---

## Algorithm 3 Simulation loop

---

```
1: for all particles do
2:    $p_{old} \leftarrow p$ 
3:    $v \leftarrow v + f_{ext} \cdot m^{-1} \cdot dt$ 
4:    $p \leftarrow p + v \cdot dt$ 
5: end for
6: Initialize multipliers  $\lambda \leftarrow 0$ 
7: while  $i < nbIterations$  do
8:   for all constraints C do
9:     Compute  $\Delta\lambda$ 
10:    Compute  $\Delta p$ 
11:     $\lambda \leftarrow \lambda + \Delta\lambda$ 
12:     $p \leftarrow p + \Delta p$ 
13:   end for
14:    $i \leftarrow i + 1$ 
15: end while
16:  $v \leftarrow (p - p_{old})/dt$ 
```

---

# 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 deformable ball interaction with ground
- 🔍 Implementation of Rayleigh dissipation potential (for damping) and generic constraint