A Position Based Material Point Method

Chris Lewin
clewin@ea.com
SEED - Electronic Arts (EA)
UK

ABSTRACT

The explicit Material Point Method (MPM) is an easily implemented scheme for the simulation of a wide variety of different physical materials. However, explicit integration has well known stability issues. We have implemented a novel semi-implicit compliant constraint formulation of MPM that is stable at any time-step while remaining as easy to implement as an explicit integrator. We call this method Position Based MPM (PB-MPM). This work significantly improves the utility of MPM for real-time applications.

ACM Reference Format:
Chris Lewin. 2024. A Position Based Material Point Method. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks (SIGGRAPH Talks '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3641233.3664323

1 INTRODUCTION

MPM is a versatile simulation approach [Jiang et al. 2016] in which particles equipped with a deformation tensor interact through a background grid. In computer graphics, it is usual to use it in combination with APIC [Jiang et al. 2015] deformation tracking and integrate the system with explicit or implicit integration. Explicit integration is easily implemented but suffers from severe stability problems that demand small time steps. Implicit integration via Newton's method can allow for large time steps, but is much more complex to implement and has other stability pitfalls such as the need to ensure positive-definiteness of the resulting stiffness matrix [Smith et al. 2019].

In real-time simulations for games, the primary concern is that the system should remain stable under any user input. This means that it should not act in an alarming way, or crash the executing device, despite potentially unsolvable situations like an incompressible fluid being crushed to zero volume between kinematically-controlled barriers. Additionally, it must be highly efficient: useful simulations should be able to run at twenty or more times real time rates.

In order to achieve this high standard, both rigid- and soft-body simulations in games are often carried out using constraint-based methods such as Position-Based Dynamics [Macklin et al. 2016; Müller et al. 2007]. In this approach, soft forces are viewed as the relaxation of a rigid constraint and are solved with a series of local iterations rather than a single global linearization. This gives a very high degree of stability at the cost of apparent stiffness becoming strongly coupled to how well the solver can converge. For real-time work, this is often a good trade-off.

In this work, we construct a novel iterative compliant-constraint formulation of MPM that combines the advantages of PBD with MPM. Specifically, it is unconditionally stable at any time step and can achieve similar results to explicit integration at much less computational cost. As with other MPM simulations, it trivially handles self-collision and one can implement a wide variety of material behaviours.

Algorithm 1 Position Based MPM. Only lines in red are substantially different to explicit MPM.

```
1: function UpdatePBMPM(P) ▷ P is an array of particles p_i
2:     for iteration ← 1 to iterationCount do
3:         P ← SolveConstraints(P)
4:         G ← ParticleToGrid(P)
5:         G ← GridUpdate(G)
6:         P ← GridToParticle(P, G)
7:     end for
8:     P ← IntegrateParticles(P)
9: end function
```

2 DETAILS

Our algorithm is very similar to explicit MPM and can be implemented with only a small number of changes. See [Jiang et al. 2016] for an excellent guide to standard MPM techniques. Algorithm 1 shows the high-level structure of our method. As with Lagrangian PBD, the main idea of the method is to iteratively improve a candidate displacement state by solving local constraints and combining the results. These constraints can be analogous to continuum-mechanical forces like co-rotational strain, or can express concepts like fluid incompressiblity that do not have an exact equivalent in force-based methods. In the accompanying videos we show two types of materials: incompressible liquids and co-rotational elastics.

Algorithm 2 shows the details of how we solve constraints for each of these materials. In each case we make changes to the deformation displacement $\mathbf{D}_i$ that are then reconciled through the MPM grid in the standard manner. Once iteration has finished, we integrate the particle's position and deformation gradient forward using the candidate $\mathbf{D}_i$ we have refined through the iterative process.

2.1 Liquids

To make liquids incompressible, we add hydrostatic impulses to $\mathbf{D}_i$ to drive a tracked liquid density L_i towards 1. Our method works quite well for liquid simulation because it is able to express incompressibility directly rather than through an equation of state. However, using MPM deformation tracking to compute the liquid

SIGGRAPH Talks '24, July 27–August 01, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0515-1/24/07
https://doi.org/10.1145/3641233.3664323

Algorithm 2 Constraint Solve

function SolveConstraints(P)
  for p_i in P **do**
    if p_i.isLiquid **then**
      $c \leftarrow \mathrm{tr}(\mathbf{D}_i)$
      $\mathbf{D}_{hydro} = \mathbb{I}(L_i - 1 - c)$
      $\mathbf{D}_{visc} = -(\mathbf{D}_i - \mathrm{tr}(\mathbf{D}_i)\mathbb{I})$
      $\mathbf{D}_i \leftarrow \mathbf{D}_i + \alpha_{hydro}\mathbf{D}_{hydro} + \alpha_{visc}\mathbf{D}_{visc}$
    else if p_i.isElastic **then**
      $\mathbf{F}^* \leftarrow \mathbf{F}_i\,(\mathbb{I} + \mathbf{D}_i)$
      $\mathbf{A}_{shape} \leftarrow \mathrm{PolarDecomposition}(\mathbf{F}^*)$
      $\mathbf{A}_{vol} \leftarrow \mathbf{F}^*/\det(\mathbf{F}^*)$
      $\mathbf{D}_i \leftarrow \mathbf{F}_i^{-1}(\beta\mathbf{A}_{vol} + (1-\beta)\mathbf{A}_{shape}) - \mathbb{I}$
    end if
  end for
end function

volume results in rapid volume loss if our solver does not fully converge. To combat this, we track an objective measure of how densely the liquid particles are packed using MPM grid transfers instead. The combination of our constraint solver and an objective volume measure allows the liquid to recover from total volume loss.

We can easily include viscosity in our model by additionally driving the off-diagonal elements of $\mathbf{D}_i$ towards zero. To control how much volume preservation and viscosity the fluid exhibits we can use relaxation factors α_{hydro} and α_{visc} respectively. Relaxation factors are a fairly crude way to control behaviour of a PBD system because their effect changes with the time step and iteration count used, but in this case it is not onerous to tune them to get the required look. For more refined control over material properties, an XPBD [Macklin et al. 2016] variant of MPM is very likely possible.

2.2 Elastics

We can achieve elastic behaviour by solving a co-rotational elasticity constraint for each particle. First we note that given a candidate velocity, we expect a candidate deformation gradient $\mathbf{F}^* = (\mathbf{F}_i(\mathbb{I} + \mathbf{D}_i)$ to be generated when the particle is integrated at the end of the frame. Thus, we can drive the deformation gradient towards any matrix $\mathbf{A}$ by rearranging: $\mathbf{D}_i = \mathbf{F}_i^{-1}\mathbf{A} - \mathbb{I}$. For shape preservation, we can set $\mathbf{A}_{shape} = \mathbf{R} = \mathrm{PolarDecomposition}(\mathbf{F}^*)$, where *PolarDecomposition* refers to the Polar SVD commonly used in elasticity simulation[McAdams et al. 2011].

For volume preservation, we can use $\mathbf{A}_{vol} = \mathbf{F}^*/\det(\mathbf{F}^*)$. Unlike the liquid case, these two constraints are not orthogonal. We can reconcile this by introducing an interpolating factor β so that $\mathbf{A} = \beta\mathbf{A}_{shape} + (1-\beta)\mathbf{A}_{vol}$.

For elastic particles we track $\mathbf{F}_i$ in the usual manner for MPM deformation gradients, which means its values can be essentially arbitrary. This means it can become very badly conditioned or even singular. This happens fairly commonly in our simulations because we use large time steps. To combat this, we delete elastic particles with a high condition number $\left(\mathrm{cond}(\mathbf{F}_i) > 10^6\right)$. We do not need to do this to liquid particles because they use an objective volume measure.

For more specific details of our method, please see the open-source code that will accompany our talk. All simulations discussed here and in our accompanying videos were done in real-time using an AMD Ryzen Threadripper 32-core CPU, using AVX2 SIMD instructions and parallelism. For grid transfers we used MLS-MPM with quadratic B-spline weighting.

3 DISCUSSION

The main drawback of Position Based Dynamics is that behaviour depends strongly on how well the solver converges, and that convergence is dependent on the problem structure. In PBD terms, our method amounts to a Jacobi-style solver because all the constraints are updated in parallel without any information propagating between them during the constraint solve step. This leads to rather slow convergence, which manifests as artificial softness.

One iteration of our method corresponds roughly to one time step of explicit MPM in terms of computational effort. However, we are able to take much smaller time steps when using PB-MPM. For instance, in a dam break simulation of liquid our simulator was stable when running at only 30Hz with a single iteration, whereas an explicit implementation with stiffness set to maintain the same volume in the steady state required 240Hz to retain stability through the splash phase. It is possible to make explicit integration more stable by limiting the forces it can produce, but this also causes artificial softness and may be too harsh an intervention in some situations. On the other hand, PBD will be as stiff as possible without threatening the stability of the simulation.

4 CONCLUSION

We have presented a simple modification of MPM that allows for stable simulation with large time steps. This significantly improves the usefulness of MPM in games, where PBD style solvers are common. This could allow the integrated simulation of new physical material types in games, as well as allowing for much higher fidelity simulation of object destruction.

REFERENCES

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. 2015. The affine particle-in-cell method. *ACM Trans. Graph.* 34, 4, Article 51 (jul 2015), 10 pages. https://doi.org/10.1145/2766996

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 24, 52 pages. https://doi.org/10.1145/2897826.2927348

Miles Macklin, Matthias Muller, and Nuttapong Chentanez. 2016. XPBD: Position-Based Simulation of Compliant Constrained Dynamics. *Proceedings of Motion in Games 2016.* https://matthias-research.github.io/pages/publications/XPBD.pdf

Aleka McAdams, Andrew Selle, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. *Computing the Singular Value Decomposition of 3x3 matrices with minimal branching and elementary floating point operations.* technical report. University of Wisconsin - Madison. http://graphics.cs.wisc.edu/Papers/2011/MSTTS11

Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118. https://doi.org/10.1016/j.jvcir.2007.01.005

Breannan Smith, Fernando De Goes, and Theodore Kim. 2019. Analytic Eigensystems for Isotropic Distortion Energies. *ACM Trans. Graph.* 38, 1, Article 3 (feb 2019), 15 pages. https://doi.org/10.1145/3241041