V International Conference on Particle-based Methods – Fundamentals and Applications
PARTICLES 2017
P. Wriggers, M. Bischoff, E. Oñate, D.R.J. Owen, & T. Zohdi (Eds)

A CONVERGENCE ANALYSIS OF THE AFFINE PARTICLE-IN-CELL METHOD AND ITS APPLICATION IN THE SIMULATION OF EXTRUSION PROCESSES

J. LUIS SANDOVAL M.[1] AND GEORG C. GANZENMÜLLER[1,2]

[1] Fraunhofer Institute for High-Speed Dynamics,
Ernst-Mach-Institut (EMI)
Eckerstr. 4, D-79104 Freiburg i. Br., Germany
e-mail: jose.luis.sandoval.murillo@emi.fraunhofer.de, web page: http://www.emi.fraunhofer.de

[2] Institut für Nachhaltige Technische Systeme (INATECH),
University of Freiburg
Emmy-Noether Str. 2, D-79110 Freiburg i. Br., Germany
e-mail: georg.ganzenmueller@inatech.uni-freiburg.de, web page:
http://www.inatech.uni-freiburg.de

Key words: material point method, MPM, affine particle-in-cell method, APIC, mesh-free, convergence analysis, extrusion

Abstract. Simulation of extrusion processes represents a large challenge for commonly used numerical methods. In our application for example, a hot melt is extruded whilst being rapidly cooled. Under these conditions of quenching, spinodal phase separation occurs which causes the formation of a characteristic micro-structure of the extrudate, consisting of solid and liquid phases. We model this process using a variant of the Material Point Method (MPM) [4], namely the Affine Particle-In-Cell (APIC) method [13]. Its hybrid particle/grid character is advantageous for simulating both fluid and solid behavior: pure Eulerian particle methods, such as classic SPH, fail for simulating solids, particularly in tension, whereas pure Lagrangian methods generally cannot cope with large deformations caused by material flow. APIC improves upon the original MPM method by using a so-called locally affine velocity representation [13] which allows the conservation of linear and angular momentum without the need of potentially unstable Fluid-Implicit-Particle (FLIP) techniques [3]. We analyze the convergence behavior of APIC and compare its accuracy against a traditional MPM variant, the Generalized Interpolation Material Point Method (GIMP).

1 INTRODUCTION

Our case of study consists of an extrusion process, where a pressurized hot melt passes through a nozzle into a long channel, where the extrudate is gradually cooled. While cooling, the melt starts separating into solid-like and fluid-like phases/structures. At the end of the process a product with solid-like characteristics is obtained. To numerically model this extrusion process we require a method able to describe material flow in form of plasticity and phase separation, both requiring of history variables that have to be advected along with the material flow.

While the liquid part of this process could be well simulated using established Eulerian mesh-based methods such as CFD [10], the solid mechanics would constitute a problem with such an approach. The same argument holds for traditional Eulerian Smooth Particle Hydrodynamics, which cannot accurately describe solid bodies. Lagrangian mesh-based methods on the other hand, are well suited for solid body mechanics, but cannot cope with the large deformations during liquid flow.

Material-Point methods however, incorporating both Eulerian mesh-based aspects and Lagrangian particle aspects, are known to be able to successfully simulate both liquid and solid behavior [9] . Broadly speaking, these methods interpolate the velocity and stress field from Lagrangian particles onto a rigid grid, and then solve the dynamic problem on the grid. At the end, the updated velocity field is transferred from the grid back to the particles. At the completion of a time step the entire grid is discarded, which underlines the meshfree character of these methods.

To avoid confusion, it is important to outline the family tree of particle/mesh methods. All of these methods derive from the original Particle-In-Cell (PIC) formulation [1], which uses a δ-distribution to interpolate values between grid and particles. The MPM method due to Sulsky [4] improved on this using linear interpolation. Today, GIMP methods [7] are commonly employed. GIMP uses higher-order polynomials, typically with a continuous first derivative of field values across neighboring cells on the grid. It is common in the literature to refer to all hybrid particle/mesh methods as MPM, regardless of the degree of smoothness of the interpolation function. We adopt the same approach here, but specify the details of the interpolation function where required.

A problem in MPM is the lack of conservation of angular momentum. Depending on how the equations of motion are integrated in time, this angular rotation is more or less severely dampened. The classic, PIC-style update uses interpolated grid velocities to advance the particles, while the more modern FLIP approach [3] interpolates only the grid accelerations back to the particles and performs the time integration in a Lagrangian manner. This approach is capable of conserving angular momentum, however, it is also severely unstable. To resolve this issue, linear combinations of FLIP and PIC updates are used in practice, which allow for a somewhat moderate dampening of rotational motion.

Recently, a promising evolution of the MPM, named APIC was published [13]. This method exhibits exact conservation of linear and angular momentum while employing true PIC-style updates. This is achieved by an enhanced interpolation scheme for the the velocity field whenever it is transferred to and from the grid. The enhancement considers gradients in the velocity field instead of assuming piecewise constant character. This greatly improves the MPM scheme, which does not conserve linear or angular momentum by default. A number of visually compelling examples in the Computer Graphics community was already published using APIC [13, 14]. However, when it comes to engineering applications, a more detailed quantification of the accuracy has to be carried out. As to the author's knowledge, no formal study has been done regarding the convergence properties of APIC. In this work we study the convergence of APIC in terms of viscosity and velocity field by means of benchmark simulations of fluid flow: Couette flow and Hagen-Poiseuille flow.

In Sec. 2.2 we will give a brief description of the general aspects of MPM and APIC. We proceed by detailing our implementation of APIC. In Sec. 3 two convergence studies by means of fluid flow benchmark simulations are presented. There, the accuracy and convergence rate of APIC applied to Couette and Hagen-Poiseuille flows is analyzed and compared to MPM. In Sec. 4 we show how APIC can successfully simulate extrusion with a strong pressure drop, while classic MPM fails due to its unstable time integration approach. We draw conclusions in Sec. 5 and also indicate remaining challenges.

2 THEORY AND IMPLEMENTATION OF MPM AND APIC

MPM is a numerical method intended for solving the differential equations of continuum mechanics. The mass flow is discretized using particles, and a background grid serves as a scratch pad to compute the gradients. As in other numerical methods for the solution of continuum mechanics problems, a solution to the equation of motion is obtained by multiplication with a set of test functions, as shown below in its integral or so called *weak* form:

$$\int_\Omega \delta\boldsymbol{u}\cdot\rho\ddot{\boldsymbol{u}}\,d\Omega + \int_\Omega \nabla\delta\boldsymbol{u} : \boldsymbol{\sigma}\,d\Omega = \int_\Omega \delta\boldsymbol{u}\cdot\rho\boldsymbol{b}\,d\Omega \tag{1}$$

ρ is the density, $\ddot{\boldsymbol{u}}$ the acceleration vector, $\nabla\boldsymbol{u}$ the deformation gradient, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, $\boldsymbol{b}$ is a field of body forces, and $\delta\boldsymbol{u}$ are the test functions (for simplicity, the boundary forces are ignored). The key idea of MPM is to solve the constitutive equations (strain-stress relationship) on the particles and the dynamics (forces and velocities) on the grid. For this purpose we need a set of test functions $w_{ip} = w(\boldsymbol{x}_p - \boldsymbol{x}_i)$, in MPM usually referred to as *kernel or weight functions*, where $\boldsymbol{x}_p$ and $\boldsymbol{x}_i$ are the positions of the particle p and the grid-node i respectively. These functions transfer information from particles to grid nodes and from nodes back to particles, The transfer from particles to grid nodes of a material property A and its gradient ∇A reads:

$$A_i = \sum_p A_p w_{ip} \tag{2}$$

$$\nabla A_i = \sum_p A_p \nabla w_{ip} \tag{3}$$

One way to define the kernel functions is to define first two sets of interpolation functions, the *particle characteristic functions* $\chi_p(\boldsymbol{x}) = \chi(\boldsymbol{x}-\boldsymbol{x}_p)$, and the *grid shape functions* $S_i(\boldsymbol{x}) = S(\boldsymbol{x} - \boldsymbol{x}_i)$. Such functions have to meet the condition

$$\sum_p \chi_p(\boldsymbol{x}) = \sum_i S_i(\boldsymbol{x}) = 1 \quad \forall \quad \boldsymbol{x} \tag{4}$$

so that any material property can be interpolated in space as $f(\boldsymbol{x}) = \sum\limits_p f_p \chi_p(\boldsymbol{x}) = \sum\limits_i f_i S_i(\boldsymbol{x})$, where $f_p = f(\boldsymbol{x}_p)$ and $f_i = f(\boldsymbol{x}_i)$ are the values of material property of particle p and node i respectively. The original MPM scheme is obtained when the characteristic function $\chi_p(\boldsymbol{x})$ is taken as the Dirac delta function:

$$\chi(\boldsymbol{x} - \boldsymbol{x}_p) = \delta(\boldsymbol{x} - \boldsymbol{x}_p) V_p \tag{5}$$

In contrast, for the GIMP method the characteristic functions can be arbitrarily chosen [7], which will determine the smoothness and accuracy of the solution. The kernel functions can then be derived as the convolution of both sets of functions in the form

$$w_{ip} = \frac{1}{V_p} \int_\Omega \chi(\boldsymbol{x} - \boldsymbol{x}_p) S(\boldsymbol{x} - \boldsymbol{x}_i) d\boldsymbol{x} \tag{6}$$

Substituting the *kernel functions* Eq. 6 into Eq. 1 we obtain:

$$\sum_p \rho_p \ddot{\boldsymbol{u}}_p w_{ip} + \sum_p V_p \boldsymbol{\sigma}_p \nabla w_{ip} = \sum_p m_p \boldsymbol{b} w_{ip} \tag{7}$$

The above line is rewritten using equations 2 and 3 to obtain a balance of forces on the grid as:

$$\dot{\boldsymbol{q}}_i = \boldsymbol{f}_i^{ext} - \boldsymbol{f}_i^{int}. \tag{8}$$

Here, $\boldsymbol{f}_i^{int} = \sum\limits_p V_p \boldsymbol{\sigma}_p \nabla w_{ip}$ and $\boldsymbol{f}_i^{ext} = \sum\limits_p m_p \boldsymbol{b} w_{ip}$ are the vectors of internal and external forces, respectively, and $\dot{\boldsymbol{q}}_i = \sum\limits_p \rho_p \ddot{\boldsymbol{u}}_p w_{ip}$ is the rate of change of momentum. For a more detailed derivation, see [8, 7].

To complete the balance of forces, the state of stress is required. The Cauchy stress tensor $\boldsymbol{\sigma}_p^{t+\Delta t}$ can be determined from the constitutive relations as a function of the deformation gradient $\boldsymbol{F}_{\boldsymbol{p}}^{\boldsymbol{t+\Delta t}}$, which is obtained from integrating the velocity gradient $\boldsymbol{F}_p^{t+\Delta t} = e^{\Delta t \boldsymbol{L}_p^t} \boldsymbol{F}_p^t$. The velocity gradient itself, $\boldsymbol{L}^t$, is obtained using Eq. 3 as:

$$\boldsymbol{L}_p^t = \nabla \dot{\boldsymbol{u}}_p^t = \sum_i \dot{\boldsymbol{u}}_i^t \nabla w_{ip} \tag{9}$$

2.1 Implementation of MPM

At the beginning of every time step at time t the following information is located at each particle: mass m_p, volume V_p^t, position $\boldsymbol{x}_p^t$, velocity $\dot{\boldsymbol{u}}_p^t$, and stress $\boldsymbol{\sigma}_p^t$. Note that with the exception of mass all quantities have a superscript indicating the time step. This is because all but the particle mass are evolved in time throughout a series of steps, see Fig. 1.

First, the mass and current state of the velocity field is transferred to the grid. To conserve linear momentum, particle velocity is multiplied by the mass.

$$m_i^t = \sum_p m_p w_{ip}, \qquad \boldsymbol{q}_i^t = m_i^t \dot{\boldsymbol{u}}_i^t = \sum_p m_p \dot{\boldsymbol{u}}_{\boldsymbol{p}}^{\boldsymbol{t}} w_{ip} \tag{10}$$

Now the deformation gradient can be computed using Eq. 9. In the next step, the stresses are calculated using the constitutive model. Here is where the practical qualities of the MPM arise , since every particle can have different material model. Thus the interaction between different phases e.g. fluids and solids can be automatically handled.

With known particle stresses, Eq. 8 yields the rate of momentum on the grid nodes, which can be integrated in time to get the new grid momenta

$$\boldsymbol{q}_i^{t+\Delta t} = \boldsymbol{q}_i^t + \dot{\boldsymbol{q}}_i^{t+\Delta t} \Delta t \tag{11}$$

This completes the dynamic evolution of the grid. Updated particle velocities are obtained according to the PIC algorithm by transferring the grid velocity back to the particles:

$$\dot{\boldsymbol{u}}_p^{t+\Delta t} = \sum_i \dot{\boldsymbol{u}}_i^{t+\Delta t} w_{ip} = \sum_i \frac{\boldsymbol{q}_i^t + \dot{\boldsymbol{q}}_i^{t+\Delta t} \Delta t}{m_i} w_{ip} \tag{12}$$

The alternative FLIP update transfers only the change in velocity from grid to particles:

$$\dot{\boldsymbol{u}}_p^{t+\Delta t} = \dot{\boldsymbol{u}}_p^t + \Delta t \sum_i \frac{\dot{\boldsymbol{q}}_i^{t+\Delta t}}{m_i} w_{ip} \tag{13}$$

In standard MPM implementations, a typical combination of 99% FLIP and 1% PIC is used to achieve a compromise between dissipation and stability. Particle positions are updated by integrating the particle velocities:

$$\boldsymbol{x}_p^{t+\Delta t} = \boldsymbol{x}_p^t + \Delta t \sum_i \frac{\boldsymbol{q}_i^t + \dot{\boldsymbol{q}}_i^{t+\Delta t} \Delta t}{m_i} w_{ip} \tag{14}$$

The grid may subsequently be deleted, since a completely new grid can be created at the next time step and there is no need for storage of the old grid values.

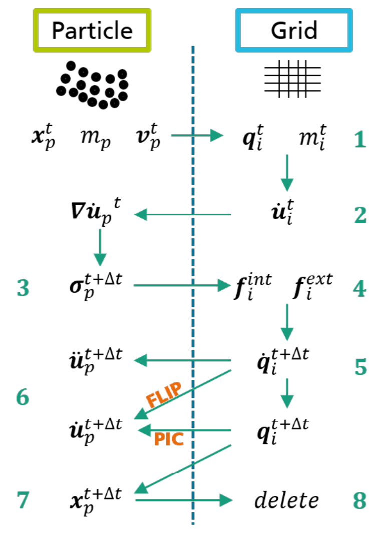


Figure 1: General algorithm of the MPM corresponding to the update-stress-first (USF) version.

2.2 APIC

MPM uses a piecewise constant interpolation in the particle/grid transfer of the velocity field. Particle mass m_p and momentum $\boldsymbol{q}_p^t = m_p\dot{\boldsymbol{u}}_p^t$ are transferred to grid nodes and then momentum is divided by mass at every node to obtain the velocity, see Eq. 10. Up to here, linear and angular momentum are conserved as long as the *kernel functions* meet the condition 4. However, particle velocity null modes (interpolated motions of particles which sum to zero on a grid node) can not be transferred to the grid due to lack of grid degrees of freedom. After the grid state of momentum is updated, velocities are transferred back to particles. In this process all velocity null modes which were not transferred to grid are lost, what causes loss of angular momentum. In a PIC transfer, old particle velocities are directly replaced with new interpolated velocities from grid, what can be seen as a filter of null modes, giving PIC its excessively dissipative characteristics. With FLIP, particle velocities are advanced, rather than overwritten, by interpolating only changes of grid velocities. This is what prevents complete loss of null modes on the particles. Thus FLIP avoids excessive dissipation but causes significant instabilities.

A solution to this problem was proposed in [13] which augments from the piecewise constant to a piecewise affine representation of particle velocity, while conserving the original PIC transfer from grid to points. The main concept of APIC is to store an additional matrix per particle which is used in the velocity transfer from particles to nodes in the form

$$\boldsymbol{q}_i^t = \sum_p m_p(\dot{\boldsymbol{u}}_p^t + \boldsymbol{C}_p^t(\boldsymbol{x}_i^t - \boldsymbol{x}_p^t))w_{ip} \tag{15}$$

where $\boldsymbol{C}_p^t = \boldsymbol{B}_p^t(\boldsymbol{D}_p^t)^{-1}$. The matrices $\boldsymbol{D}_p^t$, which is similar to an inertia tensor, and $\boldsymbol{B}_p^t$ are defined as:

$$\boldsymbol{D}_p^t = \sum_i (\boldsymbol{x}_i^t - \boldsymbol{x}_p^t)(\boldsymbol{x}_i^t - \boldsymbol{x}_p^t)^T w_{ip} \tag{16}$$

$$\boldsymbol{B}_p^{t+\Delta t} = \sum_i \dot{\boldsymbol{u}}_i^{t+\Delta t}(\boldsymbol{x}_i^t - \boldsymbol{x}_p^t)^T w_{ip} \tag{17}$$

2.2.1 Aspects of our Implementation

Certain particularities of our implementation need to be addressed. First, we used a staggered rather than a collocated grid in order to avoid the even-odd instabilities of usual MPM implementations, which manifest themselves as checkerboarding of stress fields. [2]. Further, we calculate the velocity gradient directly on the grid using finite differences and transfer the grid velocity gradient back to particles using standard interpolation. We have observed that this approach yields a smoother velocity gradient, less prone to numerical instabilities due to noise in the velocity field. Apart from these aspects, our implementation follows the work flow illustrated in Fig. 1, incorporating the velocity representation and transfer scheme of APIC.

3 CONVERGENCE STUDY

In our case of study we want to describe the velocity field of a hot melt which is extruded and passed throughout a long channel where it is cooled. Here the velocity profile of the fluid along the channel is the great interest for us. To determine how accurate we can simulate/model that flow, we set up a numerical experiment in which we can compare our simulation results with the exact solution of a benchmark problem. Hagen-Poiseuille flow is a well known problem in fluid mechanics for which an exact solution has been already derived. Next, we describe the convergence study with respect to the space discretization we carried on of both, MPM and APIC using the above mention benchmark problem.

3.1 Hagen-Poiseuille Flow

Hagen-Poiseuille flow describes the behavior of a fluid flowing along a channel formed by two stationary, infinite parallel walls separated by a distance H. Due to the no-slip boundary conditions at the channel walls a characteristic velocity profile develops and evolves over time. An (almost exact) accurately approximated solution for this problem can be obtained by means of a series method. A set of APIC-simulations of this flow were carried out using different discretization varying the amount of particles which consequently changes the particle and grid-cell size. Afterwards, we determined the convergence of the method based on the error between the numerical results with a series solution from [5].

For the simulation a square lattice was used of N by N number of particles and one particle per grid cell maintaining a height channel $H = 5$ constant. At time $t = 0$ the

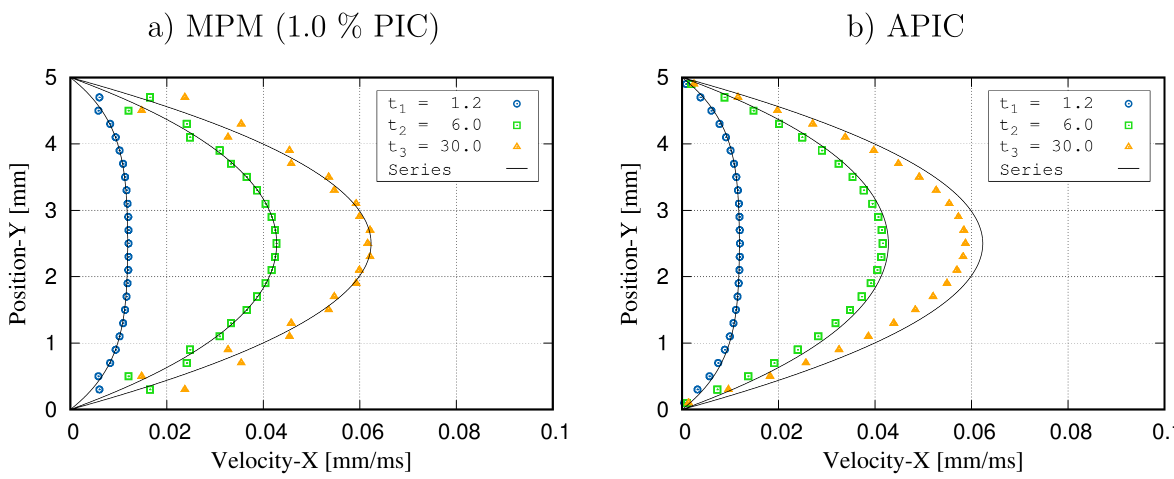


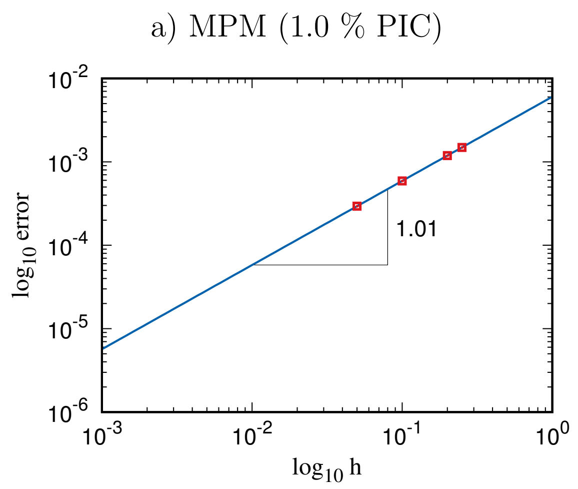
Figure 2: Velocity profile of a Hagen-Poiseuille flow at different time states. Continuous lines represent the exact solution obtained with a series method from [5]. Symbols represent the numerical results obtained with a) MPM with 1% PIC contribution (left) and b) APIC (right) using a cell size of 0.20 and particle radius of 0.10.

velocity is set to zero for all particles and for $t \geq 0$ a body force of 0.01 mm/ms^2 is set in x direction. No-slip boundary conditions with zero velocity are applied at the boundary grid nodes located at the bottom $y = 0$ and top $y = H$ of the channel. To emulate an infinitely long channel, periodic boundary conditions are applied at the left $x = 0$ and right $x = H$ sides of the lattice. The material properties used in this case were a density $\rho = 10^{-6}$ kg/mm^3, a bulk modulus $K = 1.4 \times 10^{-4}$ kN/mm^2 and a viscosity $\eta = 0.5 \times 10^{-6}$ GPa·ms.

In Fig. 2, the velocity profile at three different time plotted. The exact series solution is represented with continuous lines and numerical results using MPM with 1%PIC(left) as well as APIC (right) are represented with symbols. We observe that, although a good accuracy is obtained with MPM, significant numerical noise, or instabilities, deteriorate the velocity profile. The APIC solution, on the other hand, comes very close to the exact solution and without showing any instability.

To determine the convergence of both MPM and APIC, we measure the error of both methods obtained in a simulation of Hagen-Poiseuille flow with respect to an analytic series solution [5] using different discretization sizes. We defined the error as the velocity deviation of the numerical solution with respect to the series solution at the particle's position $e_p = v_p - v_s(x_p)$, where v_p is the is the particle position and $v_s(x_p)$ the series solution of the velocity evaluated at the particle position x_p. Then the total error of the whole simulation was computed as an L_2 norm defined as follows:

$$E = \frac{1}{N}\sqrt{\sum_{p=1}^{N} e_p^2} \tag{18}$$



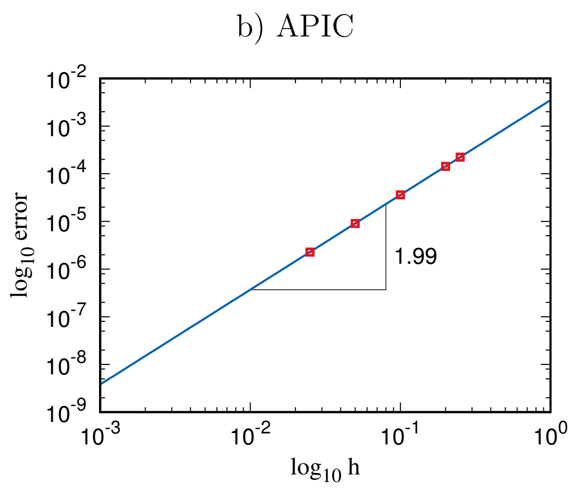


Figure 3: Convergence study for Hagen-Poiseuille flow using a) MPM with 1 % PIC contribution (left) and b) APIC (right). Simulations were performed with different cell sizes and the whole error was calculated as the L_2 norm of the difference between the series and the numerical solutions of velocities obtained at the particle positions. The logarithm of the L_2 norm of the error is plotted versus the logarithm of the cell size. The slope of the solid line which is a function fit of the error corresponds to the convergence order of the numerical method.

Fig. 3 shows a double-logarithmic plot of error versus discretization size obtained for the simulation of the Hagen-Poiseuille flow using MPM (left) and APIC (right). The solid line is the fit function of the error and its slope represents the order of convergence. The order of convergence close to unity obtained in the present study agrees well with previous studies such as [9]. Our new finding is, that APIC exhibits convergence of almost second order (1.99). This makes APIC a much better choice for studying viscous flows than conventional MPM methods.

4 EXAMPLE: EXTRUSION OF A HOT MELT

Having determined the accuracy and convergence of APIC against standard MPM, we now present a rather qualitative comparison of the simulation of an extrusion process of a highly viscous liquid. Modeling extrusion processes is a challenging task, as the strong pressure gradient of the converging flow tests the stability of the numerical code. A set of 2D simulations using APIC and MPM with different PIC-FLIP ratios were carried out to test the capabilities of both methods.

The simulation setup is based on the geometry of an extruder for meat surrogates based on vegetable proteins. It consists of a double screw extruder which moves material througho a nozzle into a long cooling channel, with a vertical flow height reduction from 25.4 mm up to 5.0 mm. The following material properties were used: Newtonian viscosity of 138.9 Pa·s, bulk modulus of 1.0 GPa and density of 1000 kg/m^3. An inflow boundary condition realizes a constant mass flow with a velocity of 2.7 mm/s. A cell size of 0.1 mm was used for the grid and a particle radius of 0.025 mm, which means 4 particles per grid cell. The simulated time was 15 seconds. Mass scaling was used to achieve a practical time step of 0.6 ms.

Fig. 4 shows snapshots from the simulations using MPM with 1% and 20% PIC and APIC. At 1% PIC, the simulation becomes immediately unstable. With 20 % PIC, the solution already becomes severely damped, as can be inferred from the homogeneous flow profile immediately after the cross section reduction. In contrast, APIC shows a much more pronounced conical convergent flow after the nozzle. Thus, APIC improves the stability of the simulation whilst preserving characteristics of the flow.

5 CONCLUSIONS AND OUTLOOK

This work compares the recently published APIC [13] variant of MPM against the more commonly used GIMP variant. APIC's improves upon GIMP by employing a velocity field which conserves linear and angular momentum. APIC also maintains the stability associated with PIC-style updates of the velocity field, without being excessively dissipative as demonstrated in [13, 15]. Here, we study the Hagen-Poiseuille problem of viscous flow through a channel. We find that APIC exhibits an increased order of convergence in this case: while normal GIMP converges with 1st order, APIC converges with second order. This makes APIC an ideal candidate for our application, which considers an extrusion process of a pressurized highly viscous melt with simultaneous thermal quenching. The challenge here is that the material behavior changes from liquid to solid, which calls for special methods such as MPM to numerically describe this process.

APIC originates from the field of computer graphics, where aesthetically pleasing visual effects are obtained by simulating physical processes. For these applications, a trade-off between computational speed and convincing physical accuracy is optimal. We are confident, however, that APIC can also be used for simulations in engineering applications, where physical accuracy is a fundamental requirement. With the present work we start to investigate the applicability of APIC for the solution of engineering systems. Future work will address the accuracy and convergence behavior of this new method for solid body mechanics problems.

REFERENCES

[1] Evans, M.W. and Harlow, F.H. ”The particle-in-cell method for hydrodynamic calculations”, *Los Alamos Scientific Laboratory*, (1957).

[2] Harlow, F.H. and Welch, J.E. ”Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface”, *The Physics of Fluids*, Vol. 8, No. 12, (1965).

[3] Brackbill, J.U., Kothe D.B. and Ruppel H.M. ”FLIP: a low-dissipation, particle-in-cell method for fluid flow”, *Computer Physics Communications*, Tech Science Press, Vol. 48, pp. 25-38, (1988).

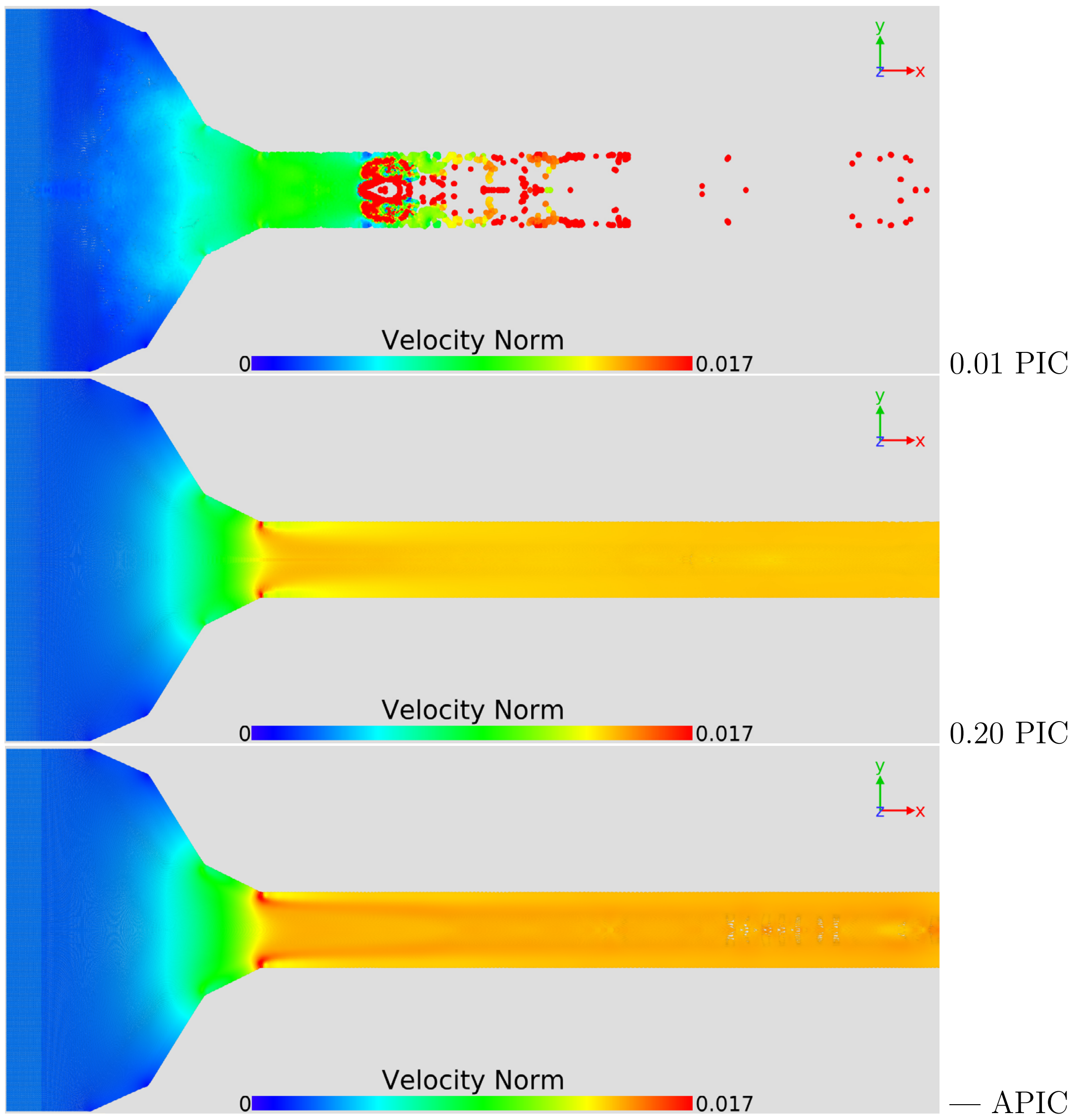


Figure 4: Snapshots of the 2D extrusion simulations using MPM with different PIC-FLIP ratios. Color scale shows the norm of the particle velocity. The snapshot for 20% PIC as well as APIC correspond to the end of the simulation at time $t = 15$ s. The snapshot of 1% PIC corresponds to the last time step simulated before the simulation broke due to instabilities.

[4] Sulsky, D., Chen,Z. and Schreyer, H.L. "A Particle Method for History-Dependent Materials", *Sandia Laboratories*, (1993).

[5] Morris, J.P., Fox, P.J. and Zhu, Y. "Modeling Low Reynolds Number Incompressible Flows Using SPH", *Journal of Computational Physics*, Vol. 136, No.1, pp. 214-226, (1997).

[6] Nairn, J.A. "Material Point Method Calculations with Explicit Cracks", *Computer Modeling in Engineering and Sciences*, Vol.4, No.6, pp. 649-664, (2003).

[7] Bardenhagen, S.G. and Kober, E.M. "The Generalized Interpolation Material Point Method", *Computer Modeling in Engineering and Sciences*, Tech Science Press, Vol. 5, No. 6, pp. 477-495, (2004).

[8] Buzzi, O., Pedroso, D.M. and Giacomini, A. "Caveats on the implementation of the Generalized Material Point Method", *Computer Modeling in Engineering and Sciences*, Tech Science Press, Vol. 31, No. 2, pp. 85-106, (2008).

[9] Wallstedt, P.C. and Guilkey, J.E. "An Evaluation of explicit time integration schemes for use with the generalized interpolation material point method", *Journal of Computational Physics*, Vol. 227, No. 22, pp. 9628-9642, (2008).

[10] Wendt, J.F. *Computational Fluid Dynamics An Introduction*, Springer, 3rd Edition, (2009).

[11] Wallstedt, P.C. and Guilkey, J.E. "A weighted least squares particle-in-cell method for solid mechanics", *International Journal for Numerical Methods in Engineering*, Vol. 85, No. 13, pp. 1687-1704, (2011).

[12] Edwards, E. and Bridson, R. "A high-order accurate particle-in-cell method", *International Journal for Numerical Methods in Engineering*, Vol. 90, No. 9, pp. 1073-1088, (2012).

[13] Jiang, C., Schroeder, C., Selle, A., Teran J. and Stomakhin, A. "The Affine Particle-In-Cell Method". *ACM Transactions on Graphics*, SIGGRAPH, (2015).

[14] Klár, G., Gast, T., Pradhana, A., Fu, C., Schroeder, C., Jiang, C. and Teran, J. "Drucker-Prager Elastoplasticity for Sand Animation", *ACM Transactions on Graphics*, SIGGRAPH, Vol. 35, No.4, (2016).

[15] Jiang, C., Schroeder, C. and Teran J., "An angular momentum conserving Affine-Particle-In-Cell method", *Journal of Computational Physics*, Vol. 338, pp. 137-164, (2017).