# Fluid Simulation Quality with Violated CFL Condition

Philipp Erler[*]
*Supervised by: Christian Hafner[†] Michael Wimmer[‡]*

Institute of Computer Graphics and Algorithms
TU Wien
Vienna / Austria

## Abstract

Fluid simulation for movies and games is still an active field of research, although the fundamental Navier-Stokes equations have been known for almost 200 years. The required computational power is extremely high, even with many approximations. Also, the visual quality still has room for improvement.

Our fluid simulation focuses on easy and fast exchange of algorithms. For example, we can switch semi-Lagrangian advection with the modified MacCormack's scheme, while the pressure projection is done with the finite element method in both variants. If the Courant-Friedrichs-Lewy (CFL) condition is met, the pressure projection can be accurate. However, this condition can be willingly violated, e.g. for performance reasons.

In our simulation with strongly violated CFL condition, we investigate how mass conservation and visual quality change with different advection methods. Also, we show the influence of a higher grid resolution and shorter time steps.

Our results show that the mass conservation is not perfect, neither with the semi-Lagrangian nor the modified MacCormack's scheme. Semi-Lagrangian advection conserves more mass in most cases. On the other hand, MacCormack's gives slightly better visual results. A shorter time step results in better fluid conservation. The visual quality can be increased by using a higher resolution and a shorter time step.

**Keywords:** Fluid Simulation, Finite Element Method, Semi-Lagrangian, MacCormack's, Courant-Friedrichs-Lewy Condition

## 1 Introduction

Animating fluids for movies and games is a very complex field. The required computational power is still extremely high, and the quality of the simulation can always be improved.

[*]ph.erler@gmx.net
[†]chafner@tuwien.ac.at
[‡]wimmer@tuwien.ac.at

Our contribution is an easy-to-understand test suite for off-line fluid simulations in MatLab[1]. Parts like advection and pressure projection can be easily exchanged. We use this application to examine the differences in mass conservation and visual quality between the semi-Lagrangian and the modified MacCormack's advection scheme with a violated Courant-Friedrichs-Lewy (CFL) condition.

There are two fundamental views for fluid simulations: Lagrangian and Eulerian. The Lagrangian approach is to simulate the fluid with particles, which is done for example in PCISPH [11]. In the Eulerian approach, the information about the fluid is given at fixed positions, usually at the nodes of a regular grid. Our application implements the Eulerian approach.

Fluids can be described with the Navier-Stokes equations. We do not care about viscosity and compressibility. Therefore, we can use their simplified version, the incompressible Euler equations [1]:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p = \mathbf{g}, \tag{1a}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{1b}$$

where $\mathbf{u}$ is the velocity vector in $m \cdot s^{-1}$ in a velocity field. $t$ is time in $s$. $\mathbf{g}$ is the sum external forces in $kg \cdot m \cdot s^{-2}$, for example gravity and forces caused by bodies interacting with the fluid. $\rho$ is the fluid's density in $kg \cdot m^{-3}$. $p$ is the pressure (force per area) of the fluid in $kg \cdot m^{-1} \cdot s^{-2}$. $\nabla$ is the gradient operator. $\nabla \cdot$ is the divergence operator.

Equation 1a describes how the fluid reacts to forces, which is much like the Newton's equation $F = m \cdot a$. Equation 1b is known as zero-divergence or incompressibility condition. When it is fulfilled, no fluid is created or destroyed, as long as the density stays constant. Solving these equations is complicated and usually requires most of the computation time [1].

The finite element method is a numerical technique to solve certain partial differential equations. We use it to solve the pressure projection equation. Simply put, we subdivide our entire simulation area into many small elements. Finding a partial solution for each element and then combining them is much easier than finding the entire

[1]`https://www.mathworks.com/products/matlab.html`

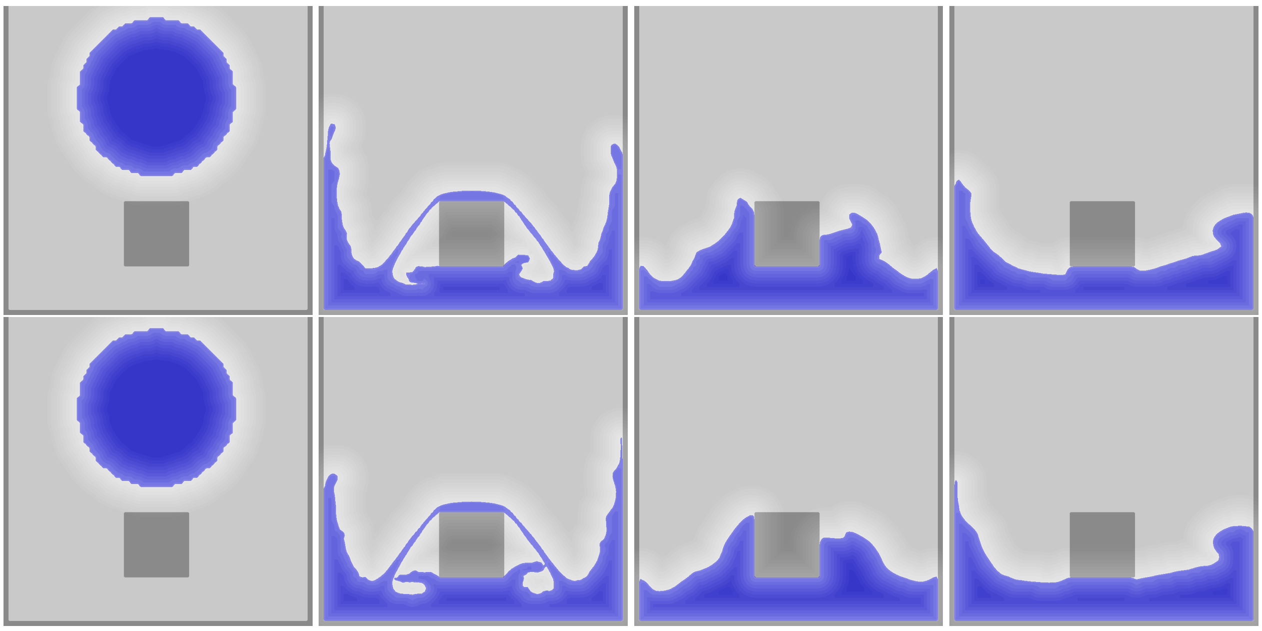


Figure 1: Single blocker scenario with 100x100 nodes at 120 FPS. The upper row is advected with semi-Lagrange and the lower row with modified MacCormack's. From left to right column: Frames 0, 60, 120, 180. Both use linear interpolation.

solution at once. The elements are primitives on a mesh. In our case, the elements are square, and their vertices are simply our grid nodes.

The Courant-Friedrichs-Lewy [3] condition is necessary but not sufficient for the stability of hyperbolic partial differential equations on a grid. When it is met, Equations 1 can be solved accurately. When the CFL condition is violated, fast moving parts of the fluid can jump through nodes and therefore ignore collisions. However, violating the CFL condition might be accepted, e.g. to reduce the computational costs. We violate the CFL condition on purpose and examine how different advection algorithms react to it.

## 2 Related Work

Stam et al. [12] introduced an unconditionally stable semi-Lagrangian advection scheme. Basically, their scheme works by backtracking from each node to its origin. It is described in Algorithm 1.

**Algorithm 1** Semi-Lagrangian Advection

```
1: for each node in grid do
2:     origin = node.position - node.velocity * deltaTime
3:     newState = interpolate currentState around origin
4: end for
```

where the current state is for example the velocity field or distance field of the last frame. Stam et al. suggest linear interpolation because higher-order interpolations cause instabilities and spline interpolation smooths details [12].

Modified MacCormack's advection scheme [10] is more accurate through an error estimation and correction step. Its main steps are:

**Algorithm 2** Modified MacCormack's Method

```
1: Semi-Lagrangian advection from the current state.
2: Backward semi-Lagrangian advection from (1).
3: res = (1) + (currentState - (2)) / 2
4: Clamp (3) with minimal and maximal values around
   the node in (1).
```

Both advection schemes are compared visually by Cohen et al. [2] showing that MacCormack's gives better results than semi-Lagrangian.

The CFL condition [3] is:

$$CFL \equiv \frac{|u|\Delta t}{\Delta x} \leq CFL_{max} \tag{2a}$$

where $CFL$ is the non-dimensional CFL number. $CFL_{max}$ is a constant depending on the solver method, which is 1 for explicit time-marching solvers. $|u|$ is the magnitude of the greatest velocity in the velocity field. $\Delta t$ is the time step and $\Delta x$ is the distance between grid nodes. With reshaping, we find the critical velocity for which the CFL condition is just met: $|u| = \frac{\Delta x}{\Delta t}$.

Pressure projection is the process of making the fluid divergence-free. We use our own FEM solver for this, which is similar to the one proposed by Guermond et al. [6]. The necessary reshaping of the incompressible Euler equations is described in Appendix A.

Surface tracking can be done with the level set method. This approach works with distance information on a grid and was originally proposed by Osher and Sethian [9]. Osher and Fedkiw gave an overview of the research on level set methods in 2001 [8].

## 3 Implementation

In this section, we describe the details of our fluid simulation. We chose to implement it in MatLab because we focus on simplicity. We want to be able to exchange algorithms easily and use the many useful functions, especially the sparse matrix solver. On the downside, we experience long calculation times, far from being suitable for real-time applications. The source code and supplemental material are available online[2].

At first, the initial state is loaded. Then, the free-slip boundary conditions are set, meaning that velocities at fluid-solid boundaries normal to the interface are set to zero. Afterwards, we start the main loop:

**Algorithm 3** Fluid Simulation Main Loop

```
 1: Apply external forces to the velocity field.
 2: Set velocities in solids to zero.
 3: Calculate the pressure field and its gradient.
 4: Subtract the pressure gradient from the velocity field.
 5: Set air velocities to the velocity of the nearest fluid
    node.
 6: Advect the distance field.
 7: Advect the velocity field.
 8: Re-initialize distance field.
 9: Draw and save current frame.
10: Save quality measures.
```

Our level set method stores the shortest distance to any fluid-air interface at the grid nodes. The distances in the air are negative. To move the interface with the fluid we simply advect the distance field with the same velocity field as for the usual advection. However, the distances within the fluid become more and more inaccurate over time. Therefore, we re-initialize the distance field regularly. We do not want the fluid-air interface to jump because of this. So, we keep the distance values around the interface.

Our simulation stores quality measures for each frame. We store the sum of the remaining divergence after pressure projection and the number of fluid pixels in the frame. This information allows us to see how the fluid volume changes.

## 4 Results

In this section, we describe our results and how to reproduce the them. We show the influence of semi-Lagrangian and modified MacCormack's advection on mass conservation and visual quality.

[2]http://www.pplusplus.lima-city.de/femfluid.html

The results are similar in all our scenarios, but the single blocker scenario shows all phenomena in the nicest way. All scenarios are simulated in a domain of 1x1 m. The visualization settings are gridCellSize = 800 / numCells, isobarWidth = 400 / numCells, and distanceRefresh interval = 1.

### 4.1 Semi-Lagrangian vs. MacCormack's

In the single blocker scenario, a circular blob of fluid falls onto a smaller blocker and then into an empty rectangular basin. When the blob collides with the blocker, the fluid gushes to the bottom left and right corners of the basin. There, the pressure values and velocities are the highest in the entire simulation. The fluid splashes in the air, falls back and slowly comes to rest.

As can be seen in Figure 1, the visual quality of semi-Lagrangian and modified MacCormack's advection is not equal but similar. The MacCormack's version features finer structures, while semi-Lagrangian has more regular structures.

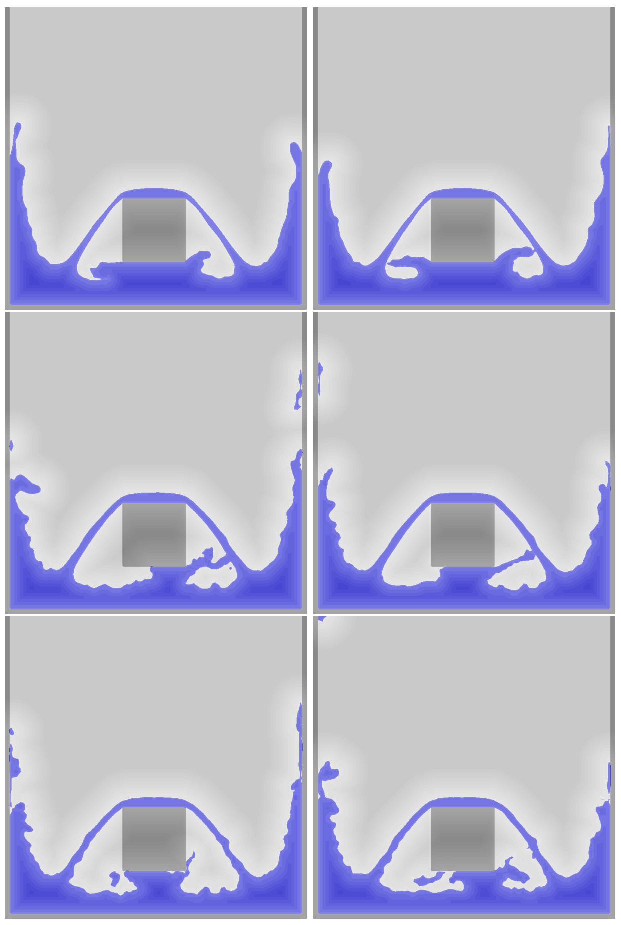


Figure 2: Single blockers scenario frame 60 at 120 FPS. The left column is advected with semi-Lagrange and the right column with modified MacCormack's. Interpolation from top to bottom row: Linear, Cubic, Spline.

Figure 2 depicts the same frame calculated with different interpolations for both advection schemes. Cubic interpolation results in more details than linear interpolation. Spline interpolation is like a slightly smoothed cubic interpolation. This fits to the assumptions by Stam et al. [12]. In our simulation, higher-order interpolations also lead to strong instabilities, which cause movement out of nowhere.

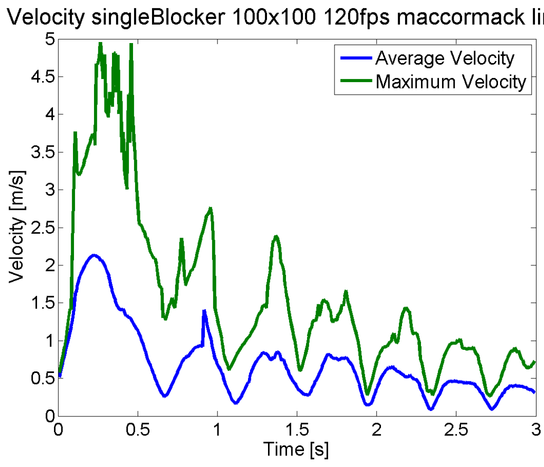


Figure 3: Maximum and average velocity of the single blocker scenario at 120 FPS advected with MacCormack's linear.

In order for the CFL condition to be met, the velocities in this simulation should never be greater than $\delta x/\delta t$, which is 1.2 m/s for 120 FPS and 100x100 Nodes in a domain of 1x1 m. Figure 3 shows the maximum and average velocities of each frame of our simulation. The maximum velocity is almost five times as high as the critical velocity for the CFL condition. Even the average velocity is too high in many frames. Therefore, we expect major instabilities in the simulation and, as a consequence, a loss of fluid.

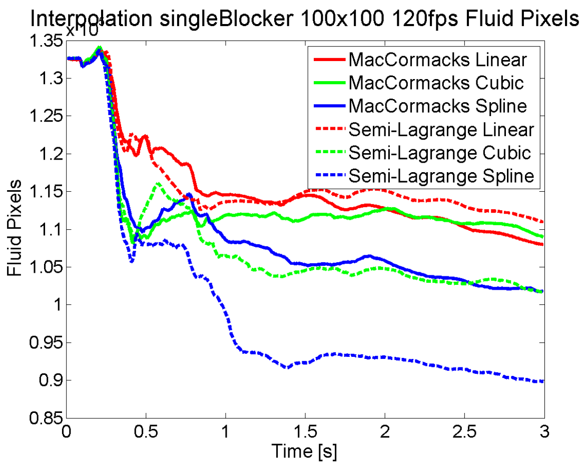
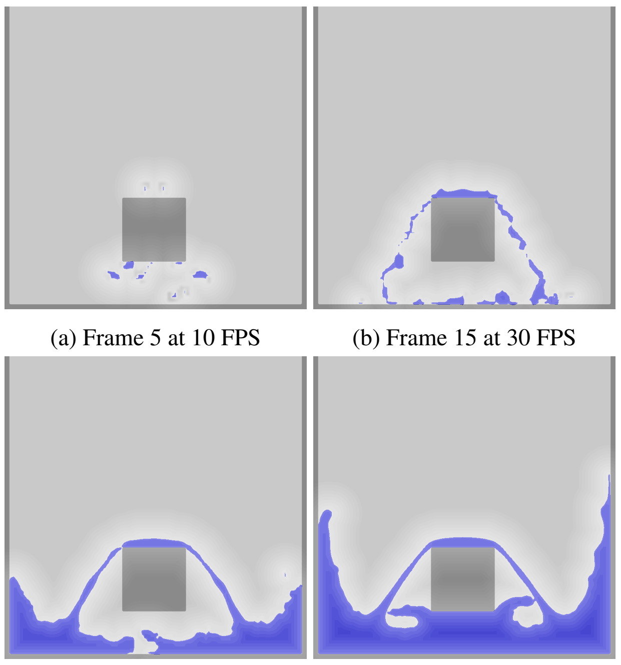


Figure 4: Number of fluid pixels of the single blocker scenario advected with different algorithms and interpolations.

The number of fluid pixels in each frame, depending on the advection algorithm and interpolation method, as can be seen in Figure 4. Most of the time, linear interpolation is the best way. Only cubic interpolation in MacCormack's is better than linear in some frames. MacCormack's spline and semi-Lagrangian cubic lose considerably more fluid than the already mentioned variant. Semi-Lagrangian with spline interpolation causes the most fluid loss.

## 4.2 Time Step vs. Resolution

In this section, we examine how the fluid simulation reacts when we shorten the time between the frames but keep the resolution constant. As shown in Section 4.1, both advection schemes give similar results, and linear interpolation is usually the best choice. Therefore, we focus here on the modified MacCormack's scheme with linear interpolation.



(a) Frame 5 at 10 FPS (b) Frame 15 at 30 FPS

(c) Frame 30 at 60 FPS (d) Frame 60 at 120 FPS

Figure 5: Single blockers scenario with 100x100 nodes at 0.5 seconds using MacCormack's with linear interpolation.

Figure 5 illustrates the state of the single blockers scenario after half a second. Figure 6 shows the fluid pixels over time. The results clearly indicate that a shorter time step reduces fluid loss. The influence on visual details is difficult to judge. Either way, there is no obvious improvement. In short, more calculation effort conserves more fluid.

Next, we increase the resolution. We can either keep the time step constant or we can shorten it to keep the critical velocity of the CFL condition constant.

When we increase the resolution of the grid but keep the time step constant, we receive the results shown in Fig-

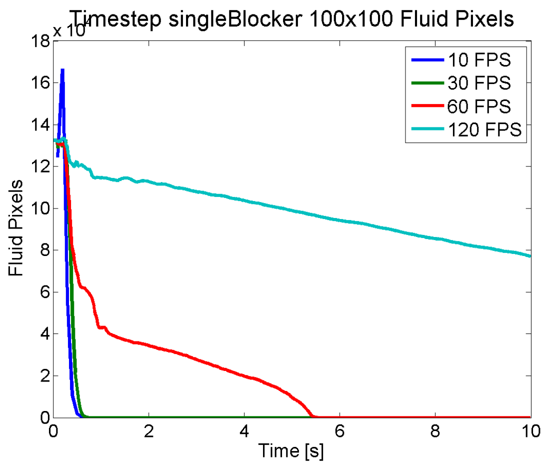


Figure 6: Fluid pixels of single blockers scenario with 100x100 nodes.

ure 7. It is not too visible here, the increased resolution results in more visual details. Also, more fluid is lost, as can be seen in Figure 8. In extreme cases, the increased fluid loss can impact the overall visual quality. In short, more calculation effort has a negative impact on fluid conservation but may improve the visual quality.

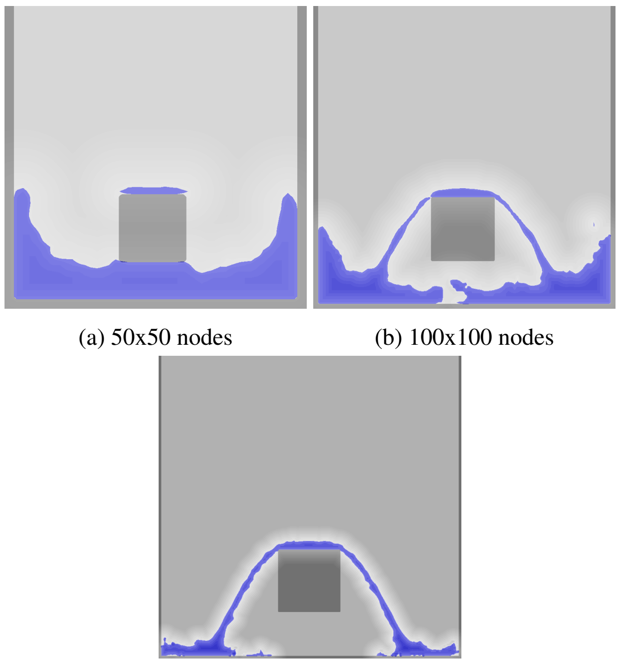
When we increase the resolution of the grid and shorten the time step, we receive the results shown in Figure 9. Obviously, with an increased resolution more visual details appear. At the upper corners of the blocker, one can see that the fluid there is less irregular, with a higher resolution and a shorter time step. Also, more fluid is conserved, as can be seen in Figure 10. In short, a greater calculation effort results in better visual quality and more fluid conservation. Also, we can conduct that the visual quality and fluid conservation does not correlate with the critical velocity of the CFL equation.

## 5 Conclusion and Future Work

Our results show that semi-Lagrangian and modified MacCormack's advection schemes are similar in terms of visual quality and fluid conservation, as long as linear interpolation is used. MacCormack's gives slightly better visual results and does not loose much more fluid with cubic interpolation.

For our simulation, with the CFL condition being strongly violated, we found the following rules of thumb: To increase the visual quality a fluid simulation, increase the grid resolution and shorten the time step. To increase the fluid conservation, focus on a shorter time step. Avoid increasing the grid resolution without adapting the time step, or the fluid will be lost faster.

For future research, we want to increase the accuracy of our simulation, for example by reducing the numerical diffusion. We will compare our current results with other advection algorithms, such as the one by Lintine et al. [7].



(a) 50x50 nodes (b) 100x100 nodes

(c) 200x200 nodes

Figure 7: Single blocker scenario after 0.5 seconds.

In addition, we intend to compare the current pressure projection using FEM with a finite differences approach. The level set method is another point that we could compare, e.g. to the improved version by Enright et al. [4]. We will also investigate the difference between ignoring the air completely and simulating it as a second fluid with much lower density, also in context of numerical stability.

## A FEM Pressure Projection

This appendix section explains how to transform the incompressible Euler equations into a useful form for our FEM solver. With operator splitting, we can get the pressure projection part out of the Equations 1a:

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{1}{\rho}\nabla p = 0 \tag{3}$$

With the forward Euler approximation, zero divergence of $\mathbf{u}$, and the Helmholtz theorem, we can reshape and simplify the pressure projection equations to this Poisson equation:

$$\frac{\Delta t}{\rho}\nabla^2 p = \nabla \cdot \mathbf{w}. \tag{4}$$

where $\mathbf{w}$ is the 'old' velocity field. $\nabla^2$ is the Laplace operator, which is the divergence of the gradient.

We solve this Poisson equation 4 with the finite element method (see Section 3). The result is the current pressure field. We calculate the pressure gradient with finite differences. We use central, forward, and backward differences to avoid calculations with the non-defined velocities

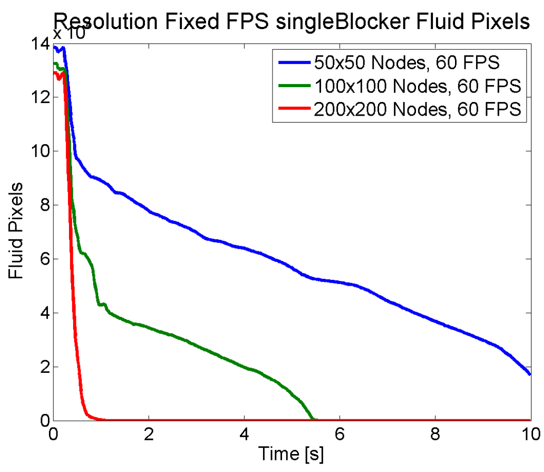


Figure 8: Fluid pixels of the single blocker scenario with different resolution and constant time step.

in solids. Then, we subtract the pressure gradient from the current velocity field and - if the solution is accurate - receive a new, divergence-free velocity field.

To use the Poisson equation in FEM, we need to transform it to its weak form and can further simplify it to:

$$\frac{\Delta t}{\rho}\int_{\Omega}\nabla p\cdot\nabla\phi\ \mathrm{d}A=\int_{\Omega}\nabla\cdot\mathbf{w}\ \phi\ \mathrm{d}A \tag{5}$$

where $\phi$ is any test function, which disappears later and is therefore not exactly specified. The weak form is an approximation that is only accurate 'on average'. The averaging is done by multiplying the weak form with the test function and integrating. We do the integration with the Gaussian quadrature. For this, we need to to get the node values everywhere in an element, which we do with bilinear interpolation. $\Omega$ is the is the fluid domain.

We create a 'local' 4x4 matrix for each element, which represents the pressure influence of each node on each node. We calculate the elements of the 'local' matrices with the Gaussian quadrature. With reshaping and Gaussian quadrature, we can turn the weak form into a form for single elements.

The right-hand side is:

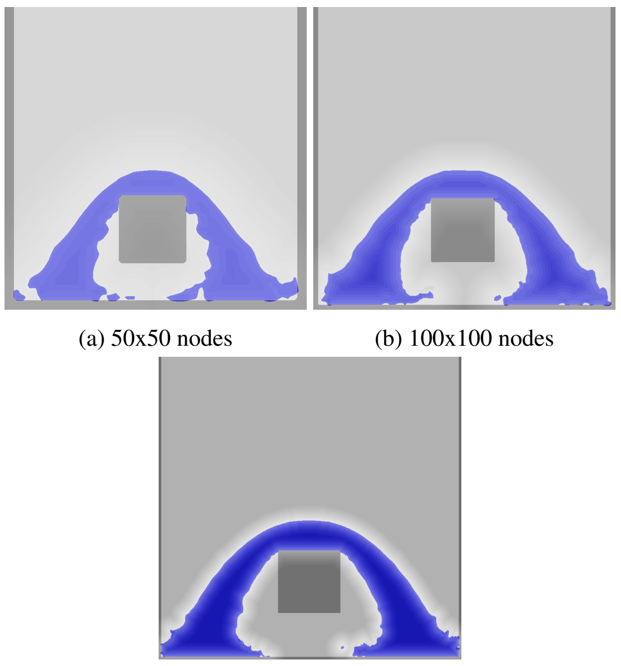
$$\mathbf{f}^e=\sum_{i=1}^{4}|\mathbf{J}(\mathbf{s}_i)|\left(\mathbf{B}(\mathbf{s}_i)^T_{|1|}\tilde{\mathbf{w}}(\mathbf{s}_i)^e_x+\mathbf{B}(\mathbf{s}_i)^T_{|2|}\tilde{\mathbf{w}}(\mathbf{s}_i)^e_y\right)\mathbf{N}\ \mathrm{d}A.$$

The left-hand side is:

$$\mathbf{K}^e=\sum_{i=1}^{4}w_i\,|\mathbf{J}(\mathbf{s}_i)|\,\mathbf{B}(\mathbf{s}_i)\,\mathbf{B}^T(\mathbf{s}_i).$$

where $i$ is the index of the element's nodes. $\mathbf{s}_i$ is the sampling position of node $i$ for the Gauss quadrature, which are

$$\mathbf{s}_{1,\dots,4}=\begin{pmatrix}\pm\sqrt{\frac{1}{3}}\\ \pm\sqrt{\frac{1}{3}}\end{pmatrix}$$



(a) 50x50 nodes (b) 100x100 nodes (c) 200x200 nodes

Figure 9: Single blockers scenario after 0.33 seconds.

The sampling weights are 1. $\mathbf{J}$ is the Jacobian matrix of the element. It encodes the partial derivatives of the shape functions with respect to global coordinates. $\mathbf{B}$ contains the partial derivatives of the shape functions with respect to global coordinates. $_{|1|}$ denotes the column vector for the x derivatives, $_{|2|}$ for y. $\tilde{\mathbf{w}}$ is the velocity at the node's sampling position. $\mathbf{N}$ is the shape function encoding the bilinear interpolation. For more mathematical details, please have a look at the elaboration by Christian Hafner[3], which is based on the work by Fuerst [5].

Boundary conditions describe how the fluid reacts at its boundaries. For example, we can ignore completely air-filled elements because its low density can only apply negligible forces on the fluid. In elements containing fluid and air, we set the influence of air nodes to zero. We keep solid objects undeformed; therefore, we can also ignore elements that are completely inside the solids. Mixed elements at the solid-fluid boundary are treated like completely fluid-filled ones. The only difference is that the velocity of the solid nodes is set the object's local velocity, which is zero in our application. The friction at fluid-solid interfaces determines if the fluid can slip along the solid. In other words, the fluid touching an object can have a different velocity than the object. When the friction is zero, the boundary condition is 'free-slip'. To make the fluid move at the same velocity as the object, the suitable boundary condition is 'no-slip'. It is also possible to have a mixture of both. We use 'free-slip', which is already included in the zero-divergence condition of the Navier-

[3]http://www.pplusplus.lima-city.de/lib/data/femfluid/FemFluid.pdf

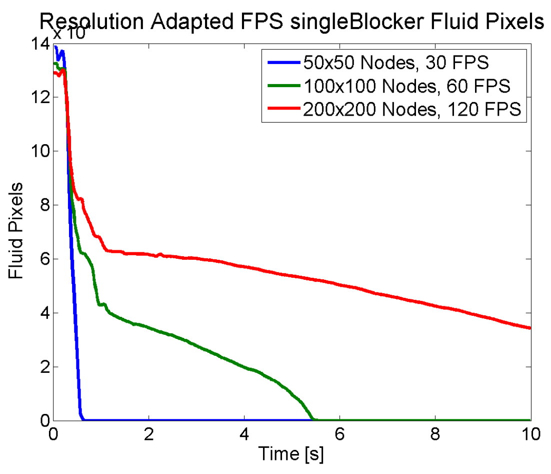


Figure 10: Fluid pixels of the single blocker scenario with different resolution and adapted time step to keep the critical velocity of the CFL condition constant.

Stokes equations 1b.

All 'local' matrices are combined to a 'global' matrix. For this, we need a global index for each node. Values from the 'local' matrices with the same global index are added in the 'global' matrix. This 'global' matrix is sparse because only few elements around the diagonal are nonzero. In our application, MatLab solves this large matrix giving us the pressure field. The Jacobi iterative method is probably the easiest algorithm to solve matrices. When using other algorithms, make sure the matrix is numerically stable, e.g. through pivoting.

## References

[1] Robert Bridson. *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press, September 2008.

[2] Jonathan Cohen, Sarah Tariq, and Simon Green. Interactive fluid-particle simulation using translating eulerian grids. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pages 15–22, New York, NY, USA, 2010. ACM.

[3] Richard Courant, Kurt Friedrichs, and Hans Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74, 1928.

[4] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83 – 116, 2002.

[5] Rene Fuerst. Real-time water simulation using the finite element method. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria, May 2013.

[6] Jean-Luc Guermond and Luigi Quartapelle. A projection fem for variable density incompressible flows. *Journal of Computational Physics*, 165(1):167 – 188, 2000.

[7] Michael Lentine, Jón Tómas Grétarsson, and Ronald Fedkiw. An unconditionally stable fully conservative semi-lagrangian method. *Journal of Computational Physics*, 230(8):2857 – 2879, 2011.

[8] Stanley Osher and Ronald Fedkiw. Level set methods: An overview and some recent results. *Journal of Computational Physics*, 169(2):463 – 502, 2001.

[9] Stanley Osher and James Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12 – 49, 1988.

[10] Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. An unconditionally stable maccormack method. *Journal of Scientific Computing*, 35(2):350–371, 2008.

[11] Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible sph. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 40:1–40:6, New York, NY, USA, 2009. ACM.

[12] Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.